CS 744: MARIUS

Shivaram Venkataraman

Fall 2022

ADMINISTRIVIA

- Midterm grades out! → 3pm today?
- Regrade requests: In-person (strongly preferred)
  - Thu: After class, Roger's OH
  - Mon: Shivaram's OH, Roger's OH
  - Tue: After class

Piazza today

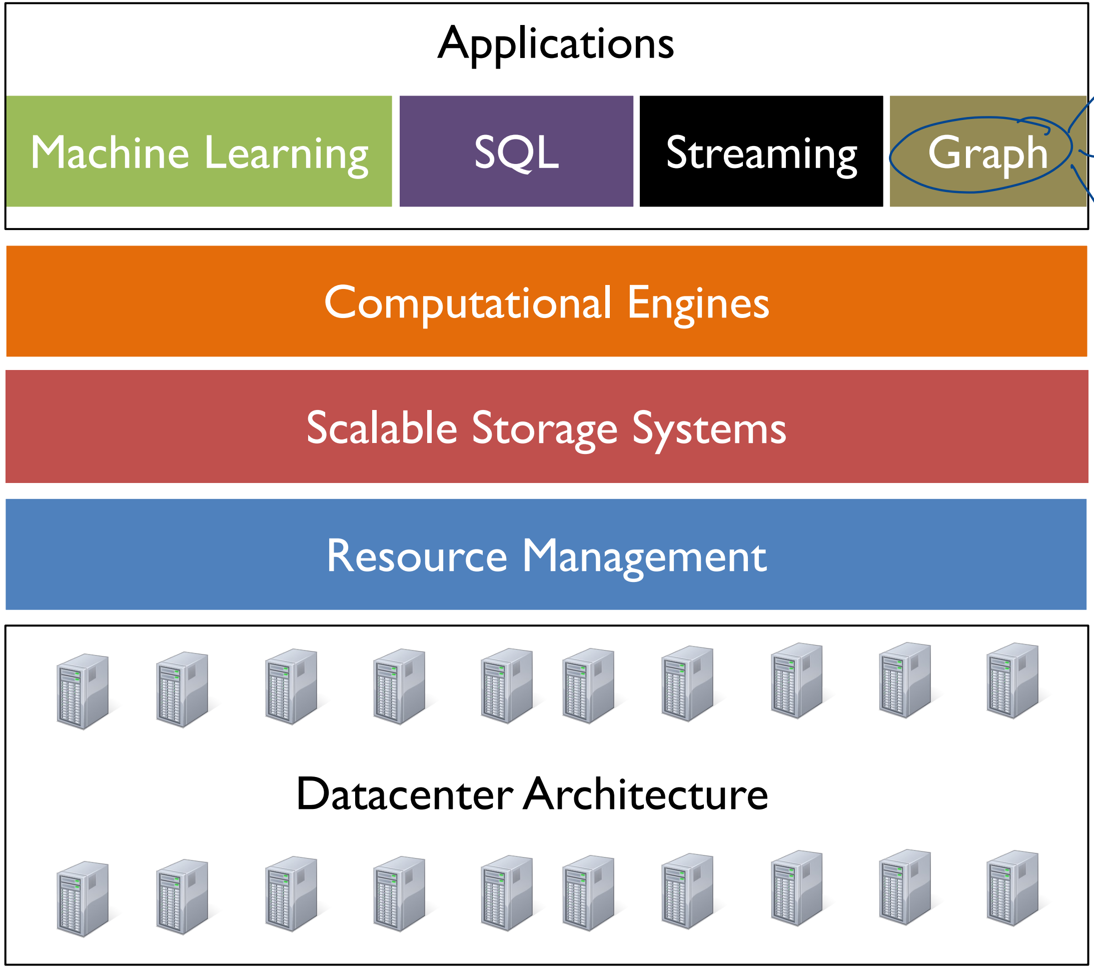
- Course Project: Check in by Nov 23th

PROJECT CHECK-INS

One page document that includes the following

- What have you done so far
- Any challenges that you have faced so far
- Your timeline (from now till end of the semester)
- Things you need help from the course staff
- Any other comments/remarks

Are you on track or/and do you need help?



Serving
Analytics
Learning

Neo4j
Powergraph

EXAMPLE: LINK PREDICTION

image, label

ML

Predicted

Task: Predict potential connections in a social network

friend reccomendation

vector representation
= embedding

Typed edge

 [0.25, 0.45, 0.30] d=3 / d=50 or 100

 [0.15, 0.85, 0.92]

…

Find K-nearest neighbors

Machine learning ?

learn embeddings
"capture "graph structure"

BACKGROUND: GRAPH EMBEDDING MODELS

Score function

Capture structure of the graph given source, destination embedding

Loss function

Maximize score for edges in graph

positive edges

Minimize for others (negative edges)

Distance function

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

e' not in graph

e is in graph

If two vertices are connected

→ embeddings to be "close"

Distance → Euclidean

Cosine Similarity

TRAINING ALGORITHM

Model = $n \times d$ matrix

→ Sparse updates to the model

1 epoch training = all the edges in the graph

SGD/AdaGrad optimizer

Sample positive, negative edges

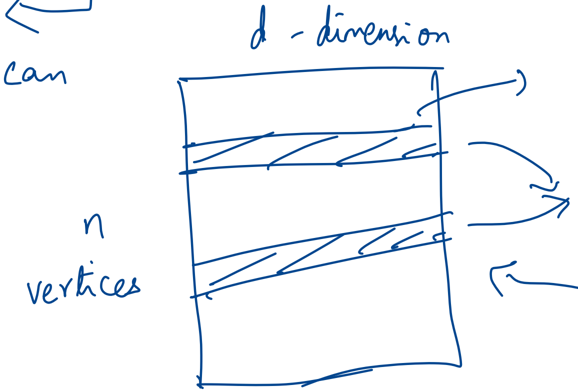
Access source, dest embeddings for each edge in batch

batch →

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

So that you can compute the score

lots of random accesses to the model



initial random

gradient

CHALLENGE: LARGE GRAPHS

model size
$d = 400$

Large graphs → Large model sizes

Example

3 Billion vertices, d = 400

$n = 3B$ vertices

Model size = 3 billion * 400 * 4 = $\boxed{\text{4.8 TB!}}$

Need to scale beyond GPU memory, CPU memory!

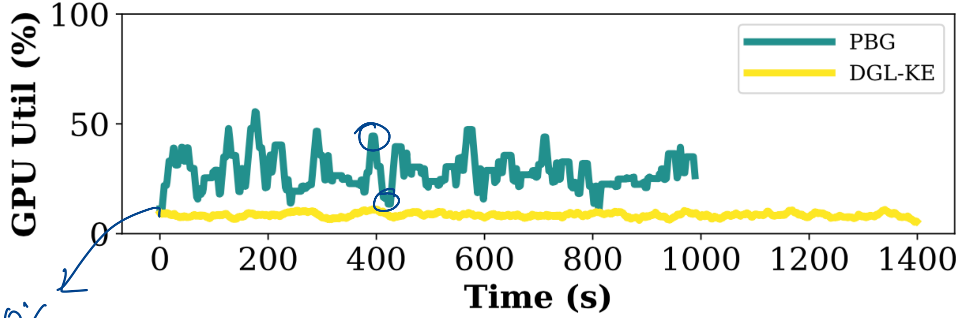
Resnet ~ 50 - 100 MB

BERT ~ 500 - 1GB

CHALLENGE: DATA MOVEMENT

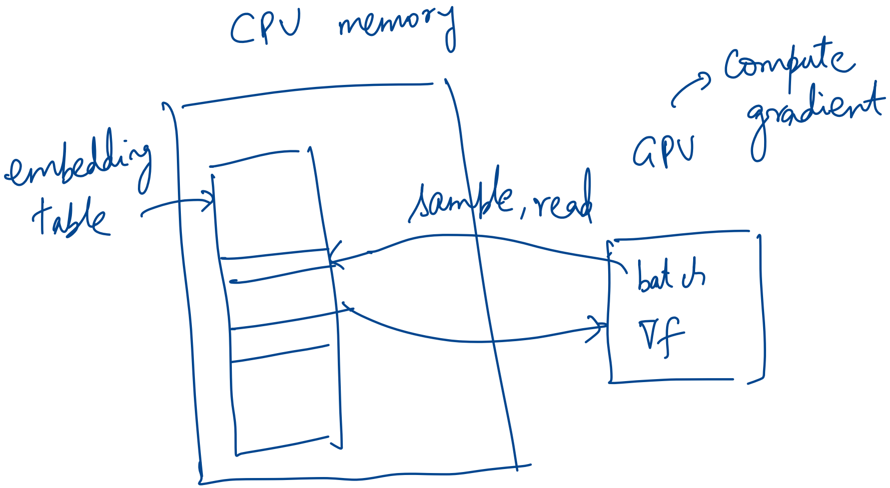
DGL-KE: Sample edges, embeddings from CPU memory → random access, PCIe for every iteration

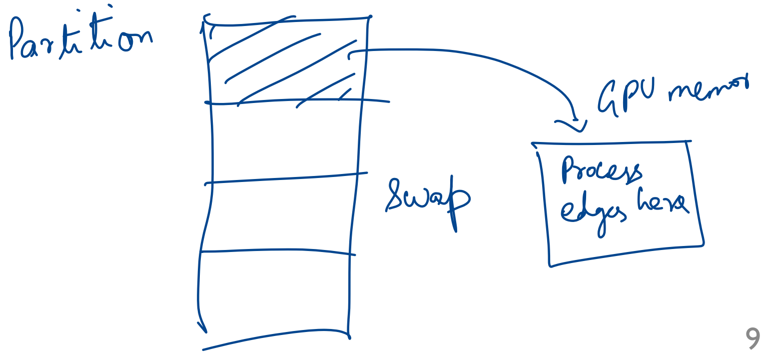
Pytorch-BigGraph: Partition embeddings so that one partition fits on GPU memory. Load sequentially



10%

One epoch on the Freebase86m knowledge graph





MARIUS

I/O efficient system for learning graph embeddings

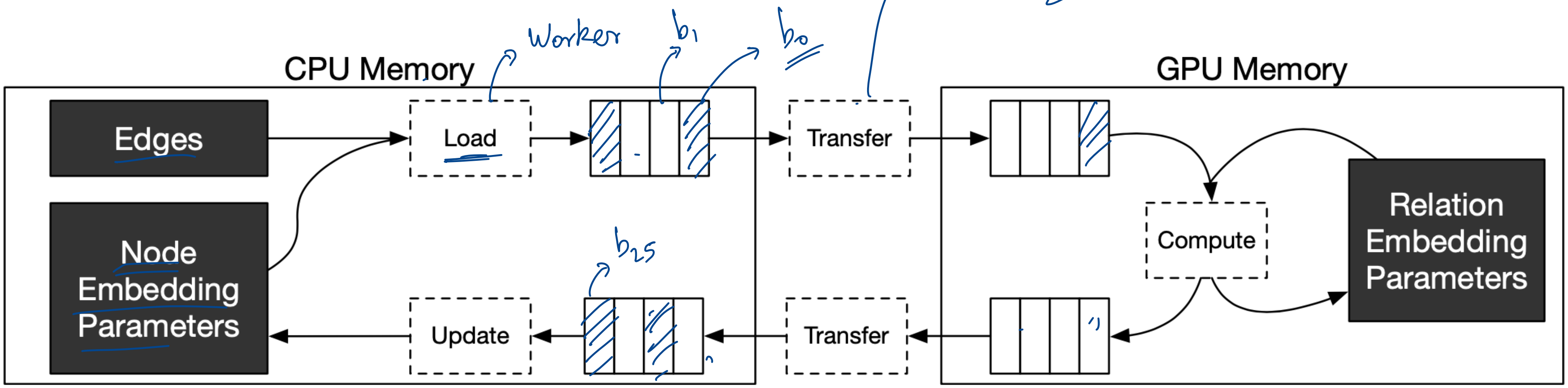
Marius Design

- Pipelined training
- Partition ordering

PIPELINED TRAINING

5 stage pipeline

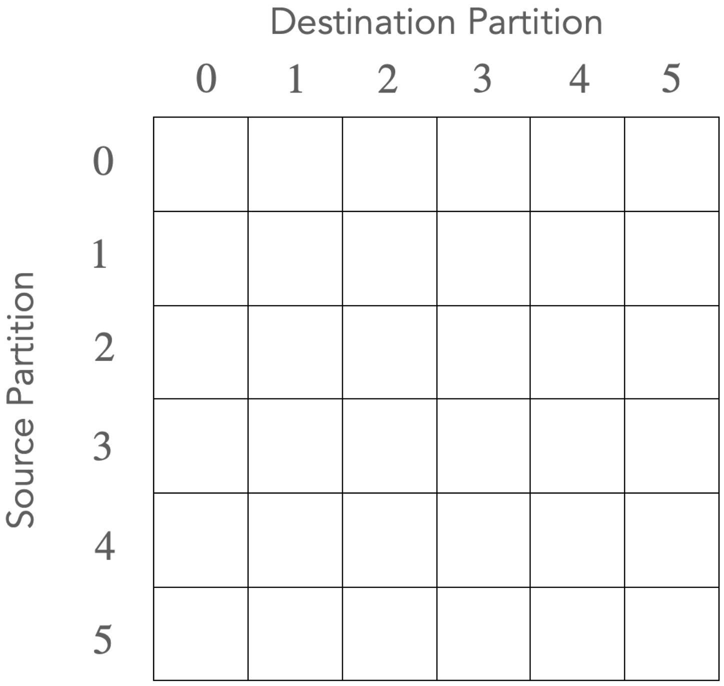
While b_0 is being transferred
b_2 is being loaded



Pipeline is as fast as slowest stage
– Queues provide back pressure

If b_0 has any embeddings in common with b_{25} then load will be stale!
→ Bounded staleness, Sparse updates

OUT OF MEMORY TRAINING

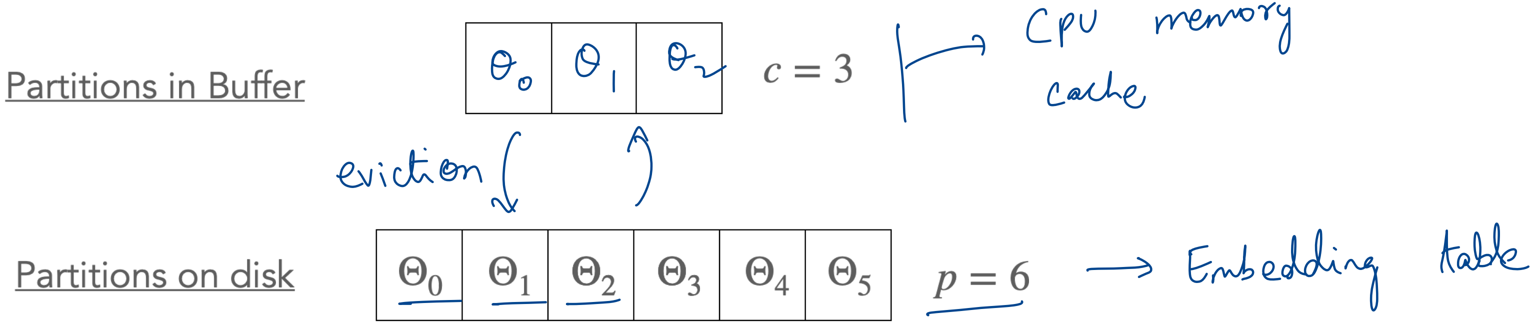


Key idea: Maintain a *cache* of partitions in CPU memory

Questions

Order of partition traversal?

How to perform eviction?

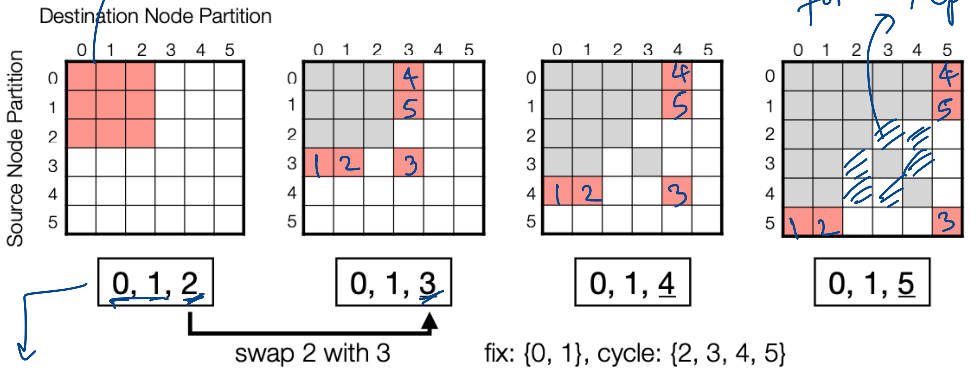


BETA ORDERING

Goal: Min number of disk swaps

adjacency matrix

all buckets for 1 epoch



cache contents

Initialize cache with c partitions

Intuition

Swap in partition that leads to highest number of unseen pairs

Achieved by fixing c-1 partitions and swap remaining in any order

⟶ very fast to get embeddings for edges with src/dest in {0, 1, 2}

At every swap
→ Keep c-1 fixed and cycle through rest

Re initialize cache and repeat

SUMMARY

Graph Embeddings: Learn embeddings from graph data for ML

Marius: Efficient single-machine training

Pipelining to use CPU, GPU

Partition buffer, BETA ordering

DISCUSSION

https://forms.gle/uNAKsPsZp56Cc1Vz9

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

Marius → batch of edges → get src / dest embedding

↳ sparse access into seq access

Powergraph → PageRank → nbr vertices

→ minimize number of remote reads for nbrs

→ Multiple machines

→ 1 - GPU + disk

better cost / perf
with Scale up
high utilization

slow and also expensive

System	Deployment	Epoch Time (s)	Per Epoch Cost ($)
Marius	1-GPU	727	**.61**
DGL-KE	2-GPUs	1068	1.81
DGL-KE	4-GPUs	542	1.84
DGL-KE	8-GPUs	277	1.88
DGL-KE	Distributed	1622	2.22
PBG	1-GPU	3060	2.6
PBG	2-GPUs	1400	2.38
PBG	4-GPUs	515	1.75
PBG	8-GPUs	419	2.84
PBG	Distributed	1474	2.02

lowest per-epoch cost

not the fastest

multiple
CPU
machines
network

If you use 8GPUs → more memory to fit embeddings, more compute
speedup

What are some shortcomings of Marius? What could the authors do to further improve the system?

BETA ordering take into account differences

load-aware scheduling

Latency is higher with pipelining ?

↳

NEXT STEPS

Next class: Distributed GNNs

Project check-ins by Nov 23th