Hello!

CS 744: MARIUS

Shivaram Venkataraman

Spring 2024

ADMINISTRIVIA

- Midterm grades ← 95% done . TODAY
- Regrade requests

- Course Project: Check in by April 16th

→ next Tuesday

PROJECT CHECK-INS

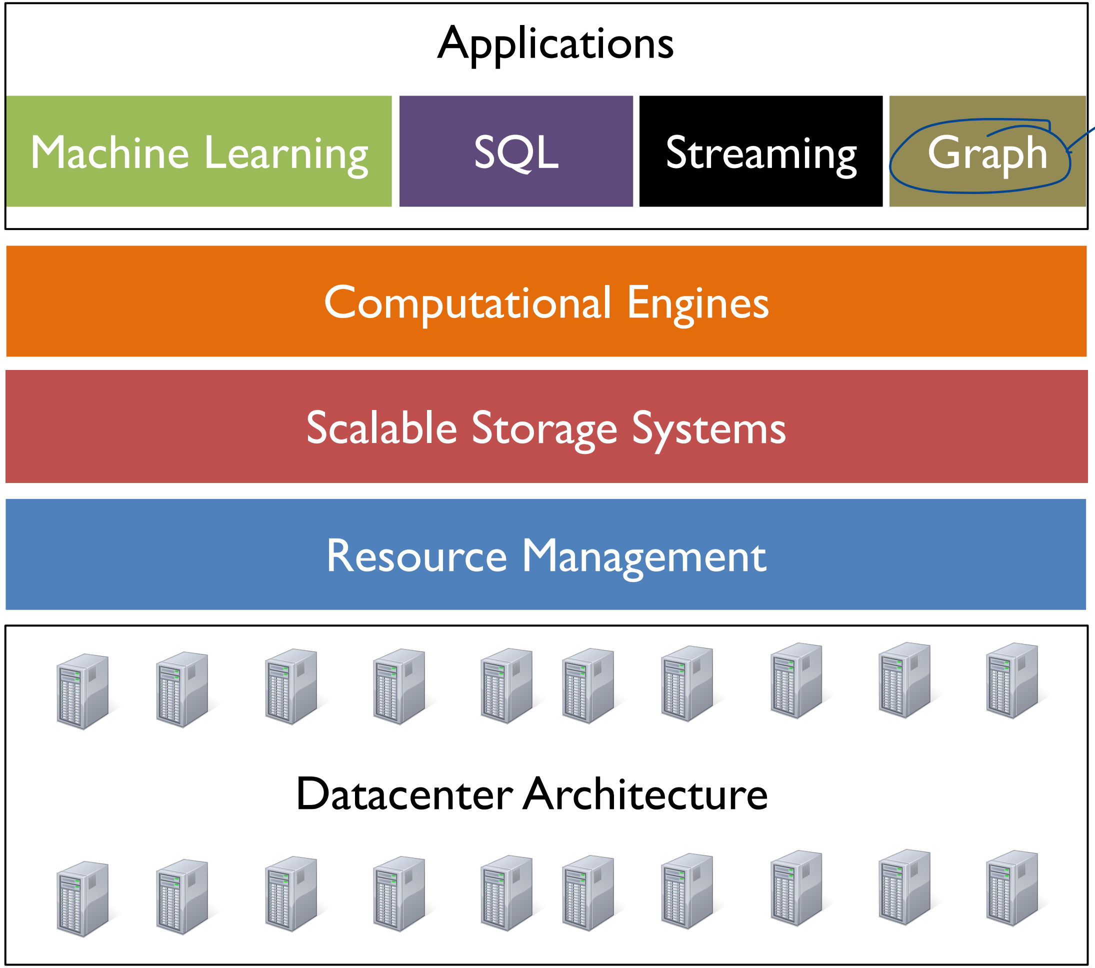
cloudlab

hardware

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

changed a lot from project introduction?



Graph Analytics

→ Page Rank
Connected Components

Learning on graph structured data ?

EXAMPLE: LINK PREDICTION

Task: Predict potential connections in a social network

edges exist in the graph

embedding ~ d vector

[0.25, 0.45, 0.30]

[0.15, 0.85, 0.92]

… learn

Find K-nearest neighbors

Predict most likely edges that should exist in this graph

BACKGROUND: GRAPH EMBEDDING MODELS

maximize score in order to learn embeddings

Score function

Capture structure of the graph given source, destination embedding

Contrastive learning

[0.1, 0.2] [0.2, 0.3]

Loss function

Maximize score for edges in graph → positive edges

Minimize for others (negative edges) → not present in graph

cosine similarity

Distance function

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

TRAINING ALGORITHM

Initialize embeddings to random vector

sample "positive" edges from the graph

SGD/AdaGrad optimizer

Sample positive, negative edges

Access source, dest embeddings for each edge in batch

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

embedding → d = size of embedding (100)

Model is this!

vertex (n) (1M)

∇g

∇g

initial embeddings

CHALLENGE: LARGE GRAPHS

Large graphs → Large model sizes

Example

3 Billion vertices, d = 400

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

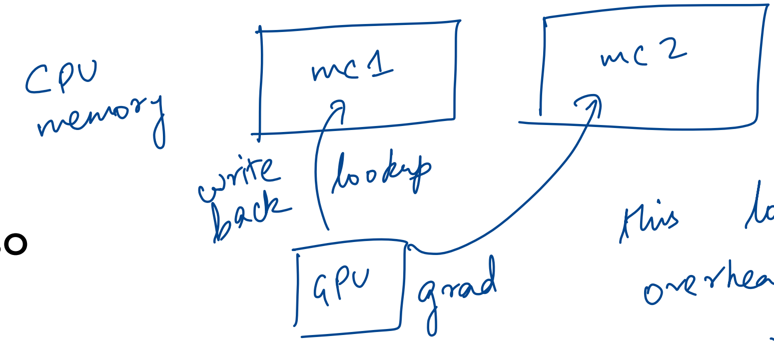
embedding table size

Need to scale beyond GPU memory, CPU memory!

CHALLENGE: DATA MOVEMENT

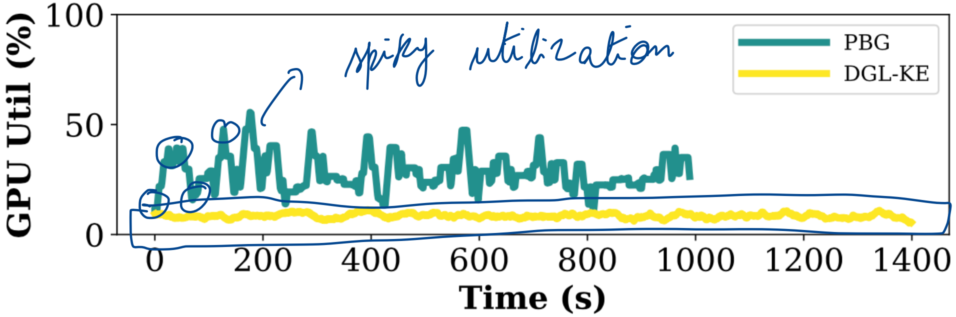
Shard emb. table

DGL-KE: Sample edges, embeddings from CPU memory



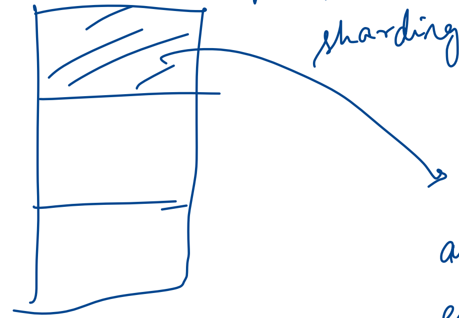
this lookup overhead is quite significant

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



One epoch on the Freebase86m knowledge graph

Partition or sharding



load block and process all edges in block

MARIUS

I/O efficient system for learning graph embeddings

Marius Design

- Pipelined training → how to keep the GPU always busy
- Partition ordering

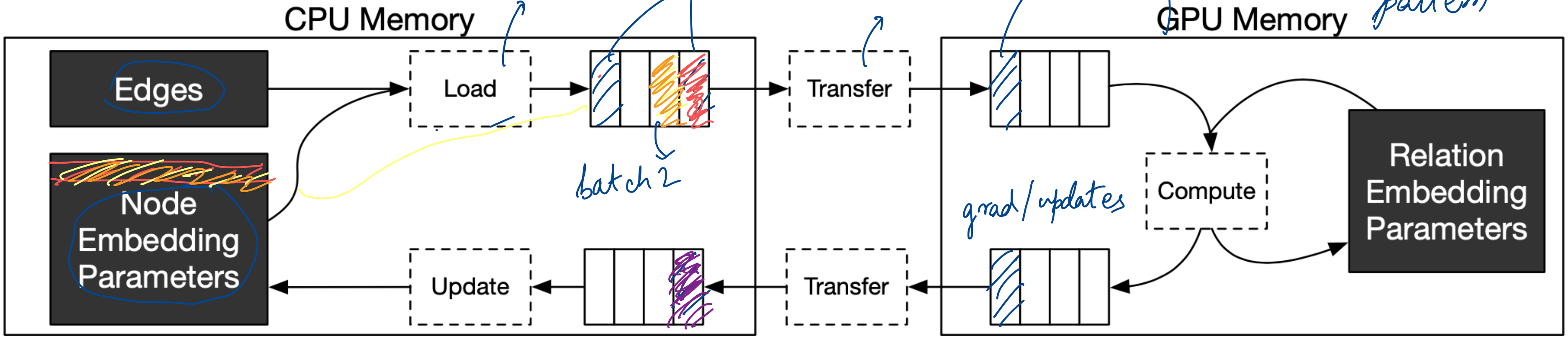
→ minimize I/O during 1 epoch

PIPELINED TRAINING

simple case
- everything fits in
CPU memory

Staleness
→ bound size
of queues between
stages
→ sparse access
pattern

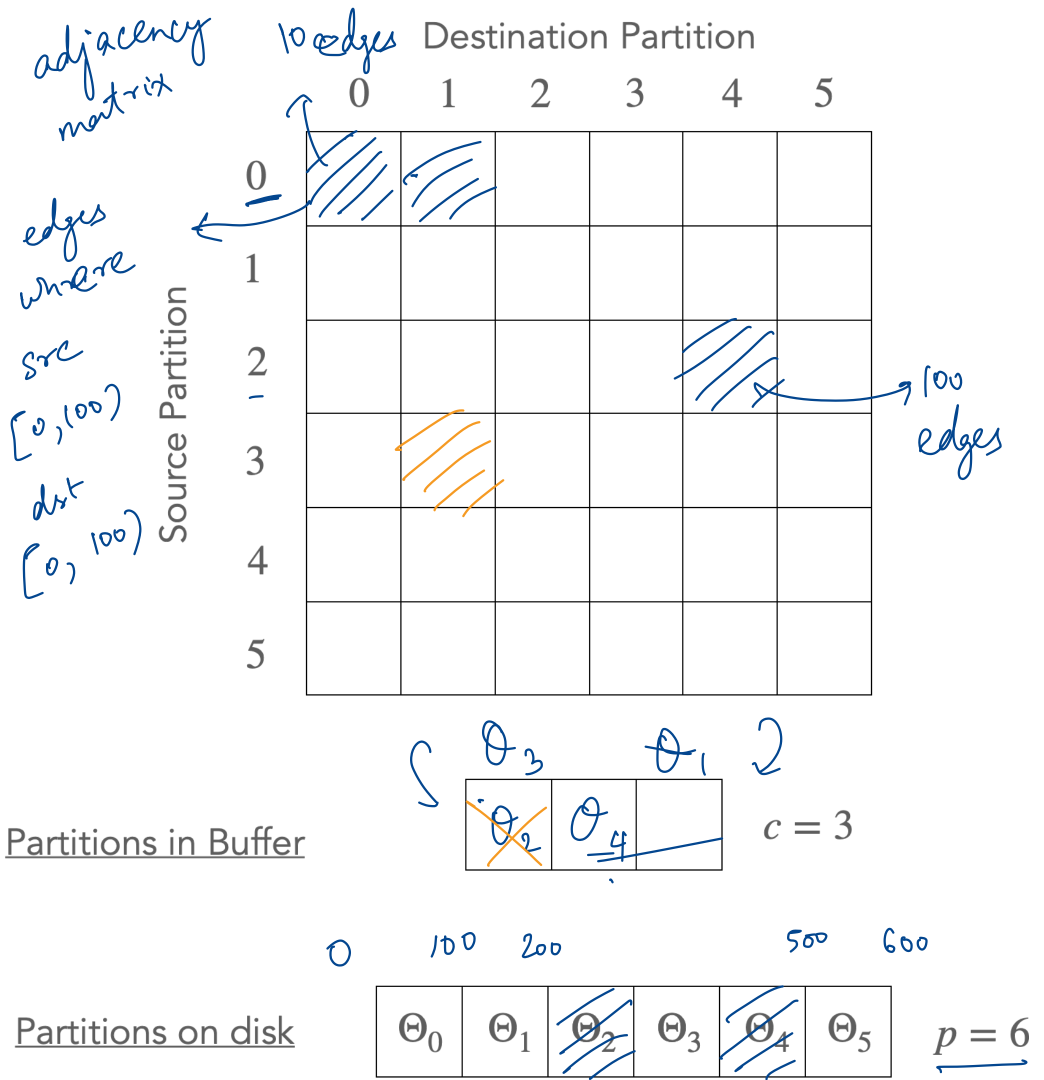
threads
batch 1
queue
threads
emb.



① sample positive, neg edges
load emb parameters
② Transfer emb, edges to GPU

③ Compute grad GPU
④ Transfer back
⑤ Apply on emb. table

OUT OF MEMORY TRAINING



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

Questions

Order of partition traversal?

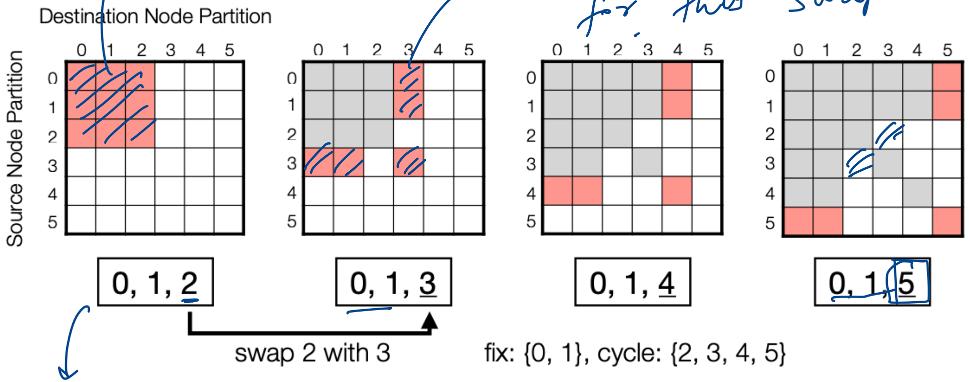
How to perform eviction?

1 pass over all edges
⇒ visit all buckets in edge partitions

BETA ORDERING

all edges can be processed

work that I can do for this swap



Initialize cache with c partitions

Swap in partition that leads to highest number of unseen pairs

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

in-memory = c partition ; c=3

More work for every swap ⇒ minimize number of swaps

SUMMARY

Graph Embeddings: Learn embeddings from graph data for ML

size of embedding table

large

→ access pattern sparse

Marius: Efficient single-machine training

→ Pipelining to use CPU, GPU

→ Partition buffer, BETA ordering

DISCUSSION

https://forms.gle/9H6dhiiSUtJU29yd7

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

Powergraph → minimizing replicas / nbrs which are remote

Marius → ordering of computation

Similarity → each edge is in only 1 bucket !!

↳ vertex may be visited many times.

high utilization → cheaper

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

cheaper?

more GPUs utilization goes down

faster than 1-GPU Marius

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

NEXT STEPS

Next class: Recommendation Models

Project check-ins next week