

Hello!

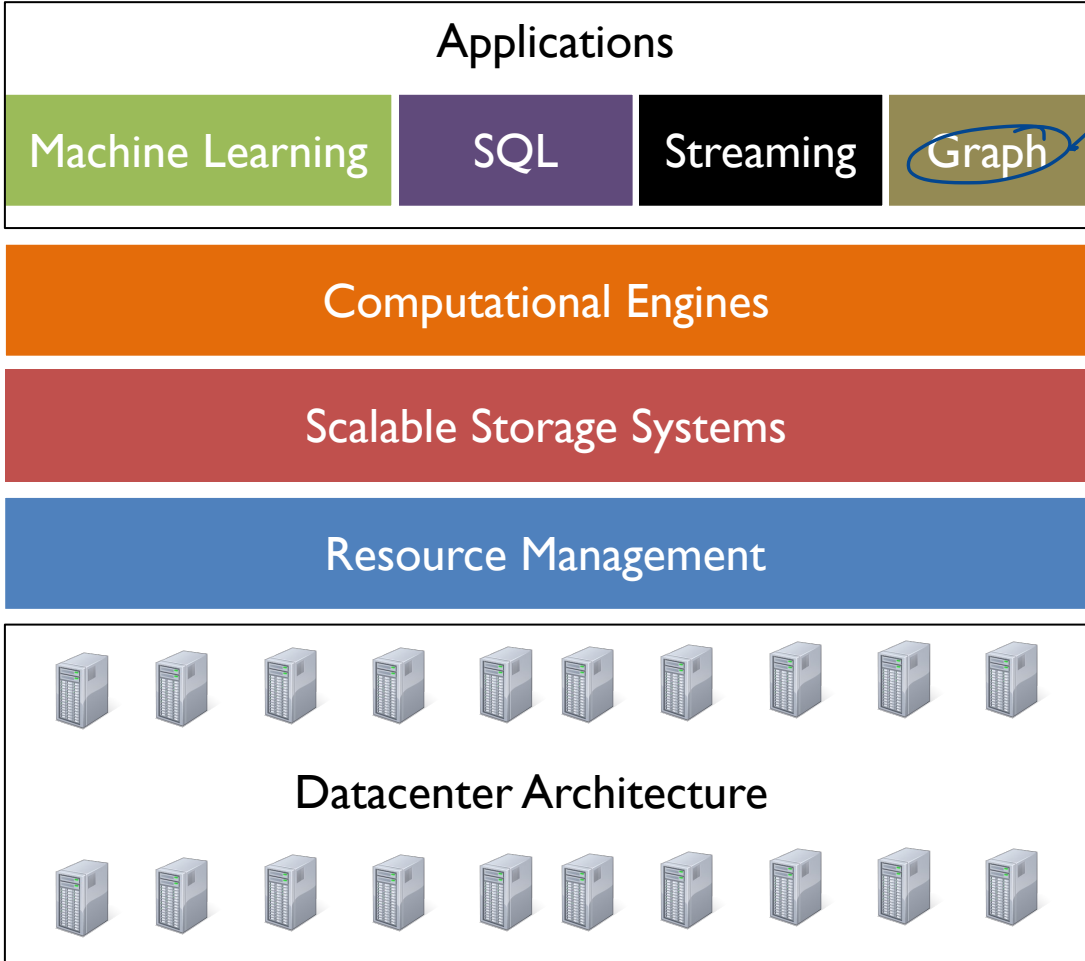
CS 744: BAGPIPE

Shivaram Venkataraman

Spring 2025

ADMINISTRIVIA

- Course Project
 - Introduction grades
 - Checkins
 - Midterm grading
- ↳ soon!
- written feedback
- schedule meetings ~ 10-15 mins
- Paper reviews → Tareq tool

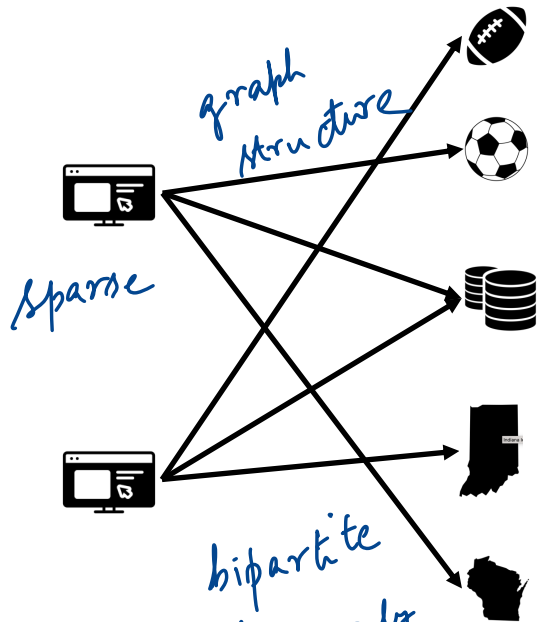


Marius
↳ ML embedding models training
Vector-DB
↳ inference
Recommendation models

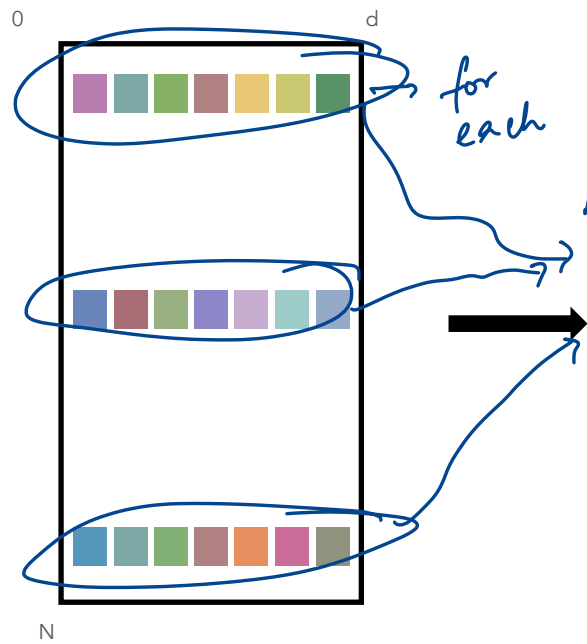
RECOMMENDATION MODELS

early 2000 → Netflix Dataset

↳ Deep Learning RM → (DLRM_s)

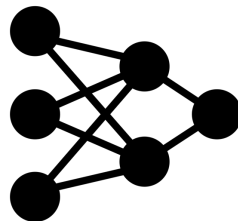


Sparse



Examples: DLRM, DeepFM, Wide & Deep

inputs to the DL model



Dense Neural Network

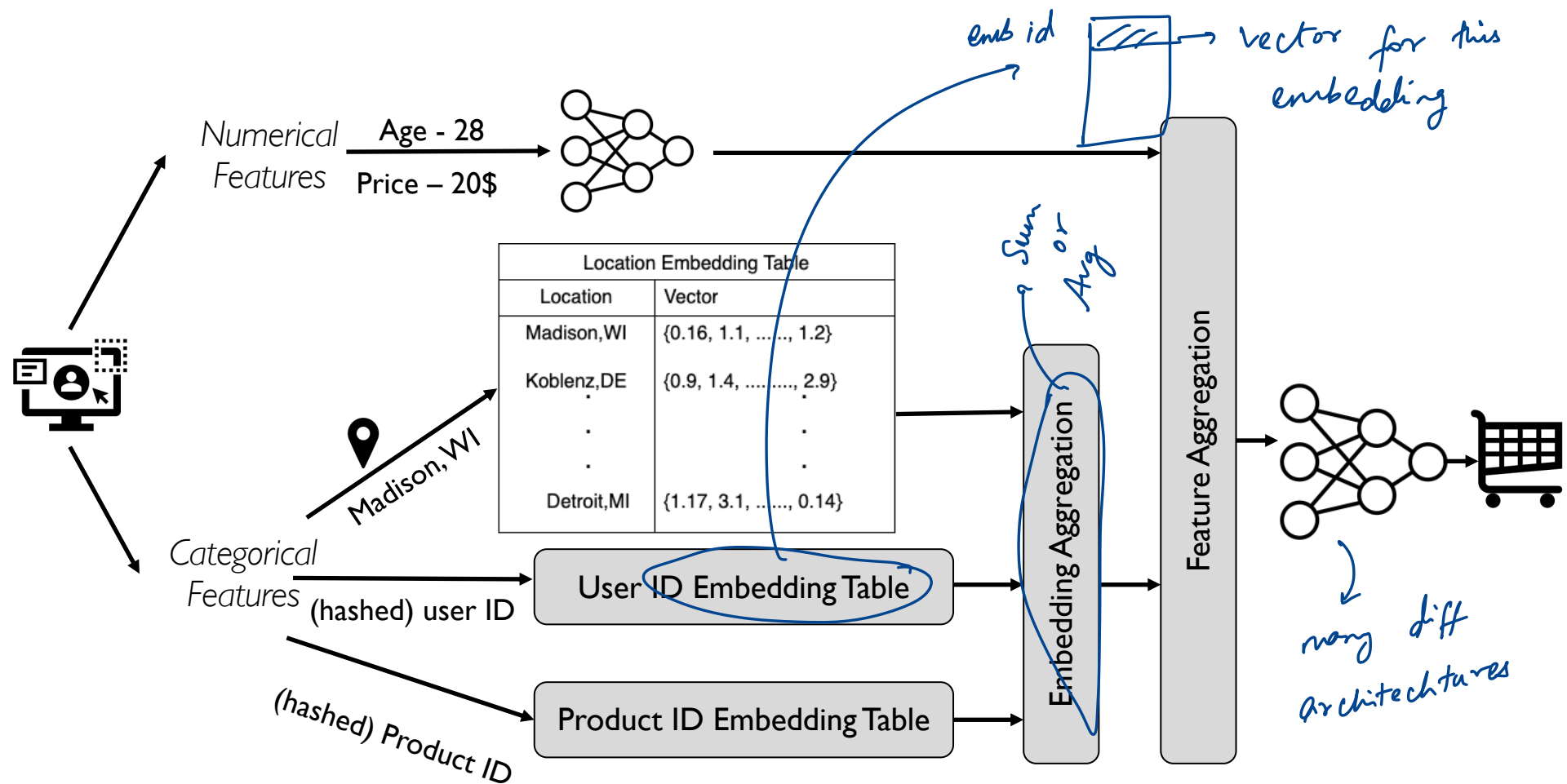
for every example

↳ a batch of clicks

Clicks → go from → Categories

1M categories

Category Embeddings



EMBEDDING TABLES

Convert categorical features to numerical features
Example: Geographic Location to a vector

Geographical Location
Embedding Table

Extremely memory intensive, could be up to TBs

UserID Embedding Table

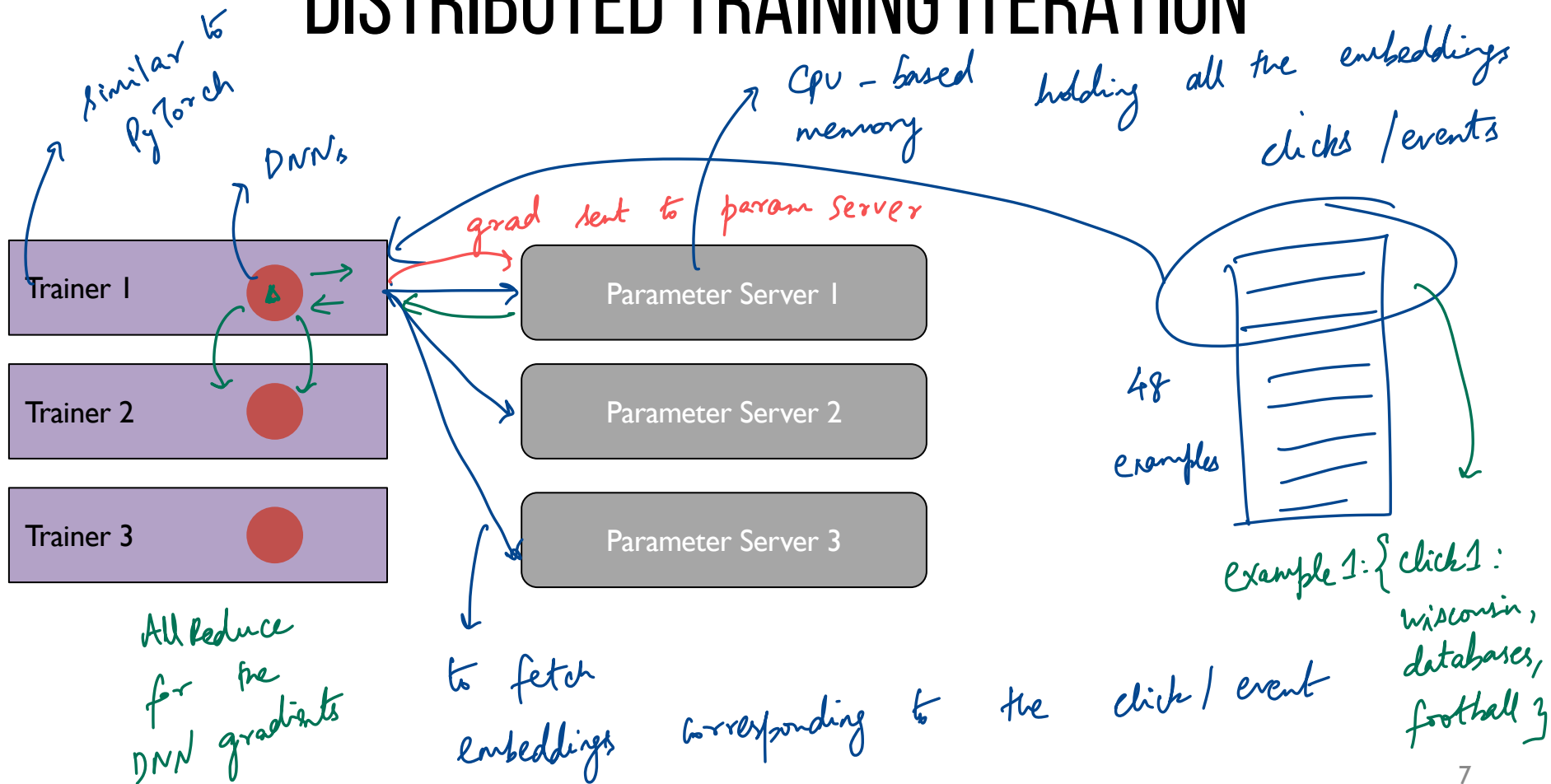
*↳ cannot fit all the embeddings in
GPU / CPU memory on single machine*

Have sparse access pattern

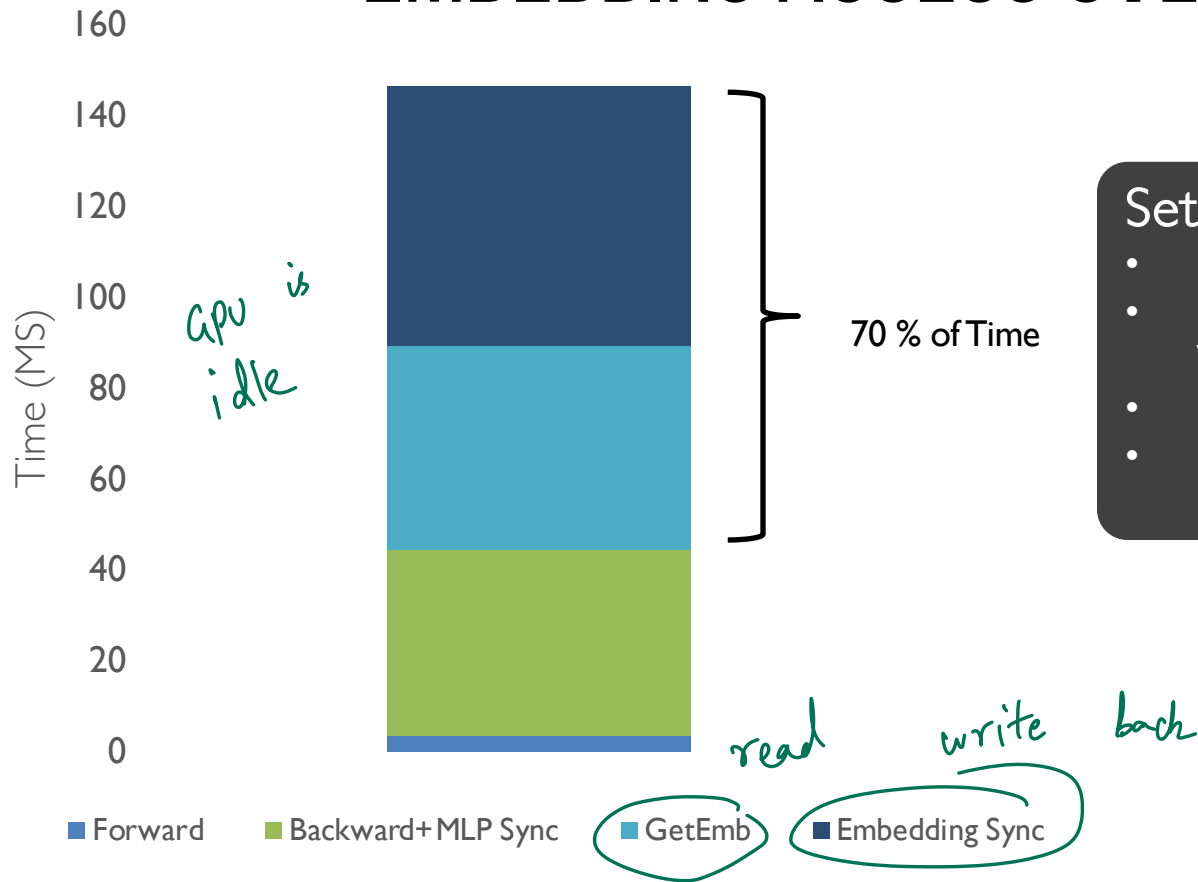
Product ID Embedding Table

*↳ only a few embeddings
are accessed by each example*

DISTRIBUTED TRAINING ITERATION



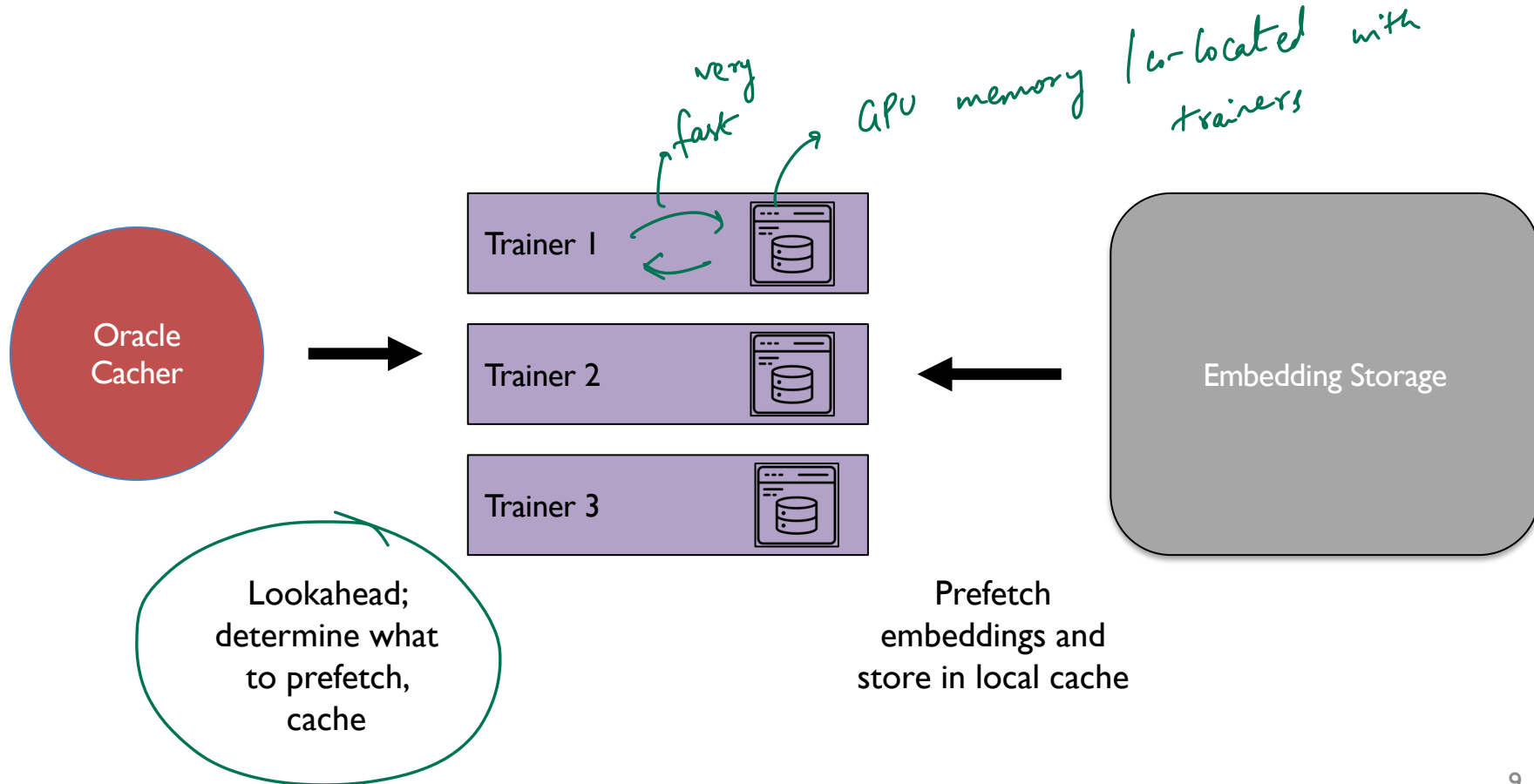
EMBEDDING ACCESS OVERHEADS



Setup:

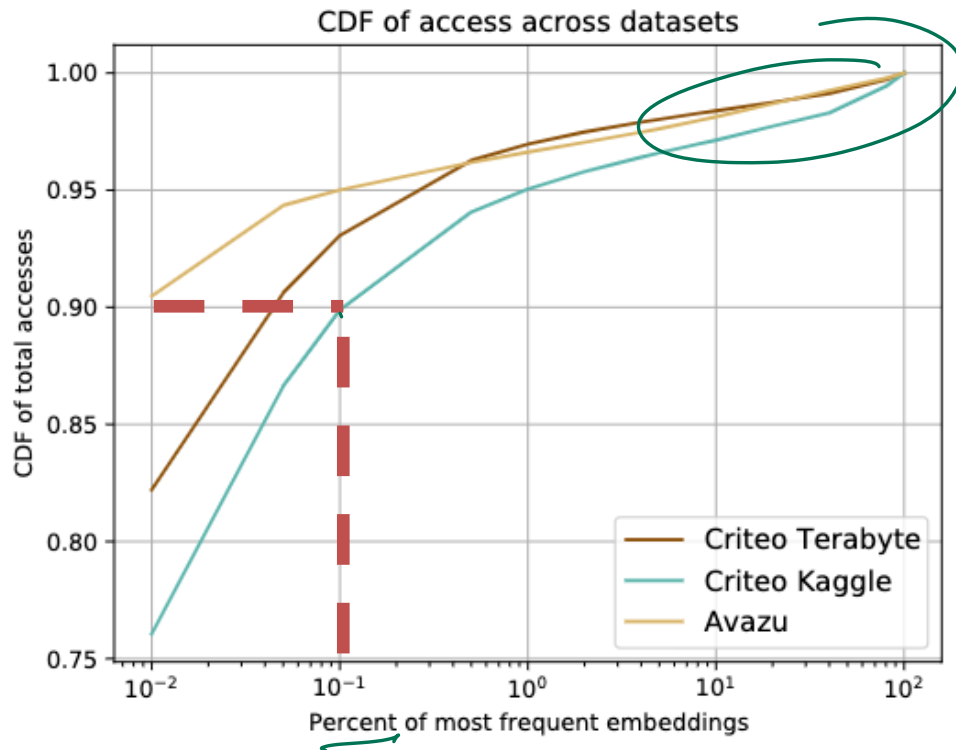
- DLRM model
- 8 trainers (p3.2xlarge EC2 instances | V100 each node).
- Batch 2048 per machine.
- Criteo Terabyte Dataset

BAGPIPE DESIGN



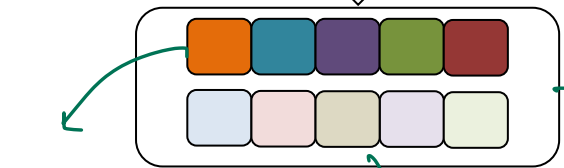
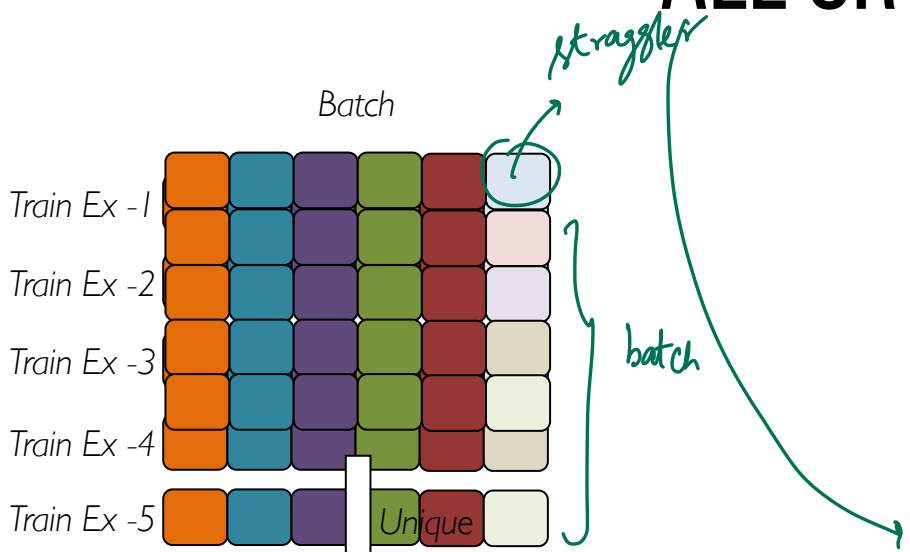
EMBEDDING ACCESS PATTERNS

sparse
very large
skew



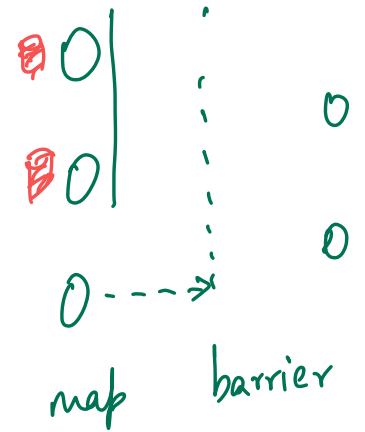
90% of
all accesses
go to
0.1% percent
of embeddings

“ALL OR NOTHING”



what will be issued by Data loader

90%



Models are trained with a batch of examples.

- For a batch only fetch unique embeddings
- Since hot embeddings are replicated, unique embeddings are comprised of long-tail accesses.

L small
↳ cache misses

LOOKAHEAD ALGORITHM

L large?
↳ too large to cache

Look at " L " next batches ahead of current batch to extract access pattern of embeddings by future batches

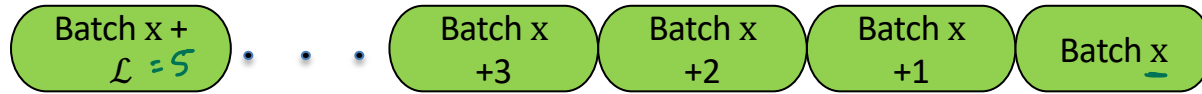
→ Bipartite graphs from clicks to categories

→ Perfect caching
Belady

Pre-fetching

Current Batch

→ if an embedding in x is used in $\{x+1 \dots x+5\}$ keep it in cache

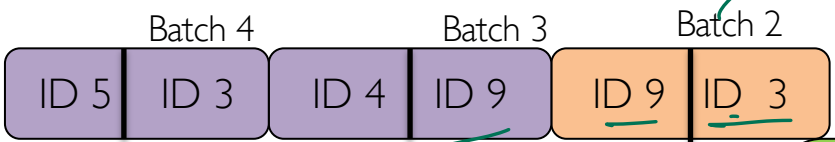


$L =$ lookahead factor

number of batches you will look ahead

→ get emb used by $x+1$ while x is being processed

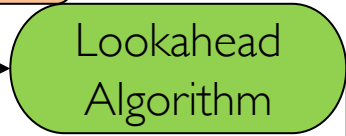
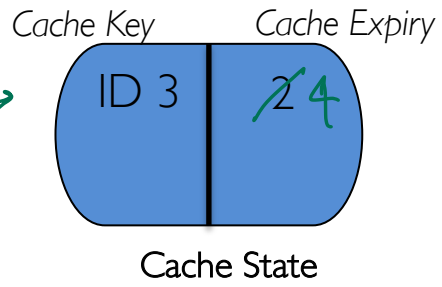
Look-ahead Value of 2, Batch size of 2



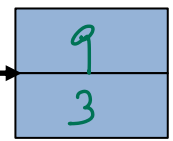
3 is used by batch 4

iteration

keep ID 3 in cache until batch 4 is done



EmbID to Prefetch



Cache EMB

Expiry



Expiration Update

set of instructions sent to trainer

LOOKAHEAD GUARANTEES

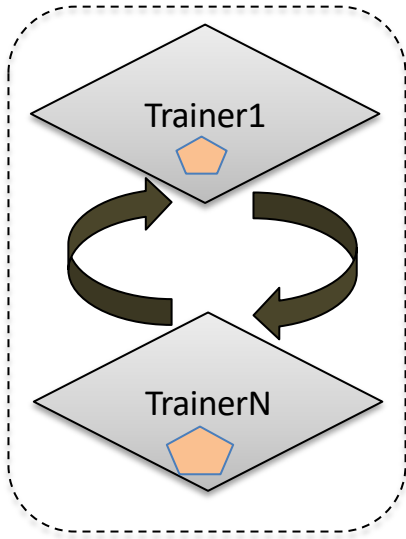
An embedding used by batch x , will either be available in cache, or no preceding batch in range $[x - \mathcal{L}, x)$ has accessed it.

Consequently, we can prefetch embeddings used by batch x , once embeddings for batch $x - \mathcal{L}$ have been updated

model will synchronous training!
have same
semantics

CACHE SYNCHRONIZATION

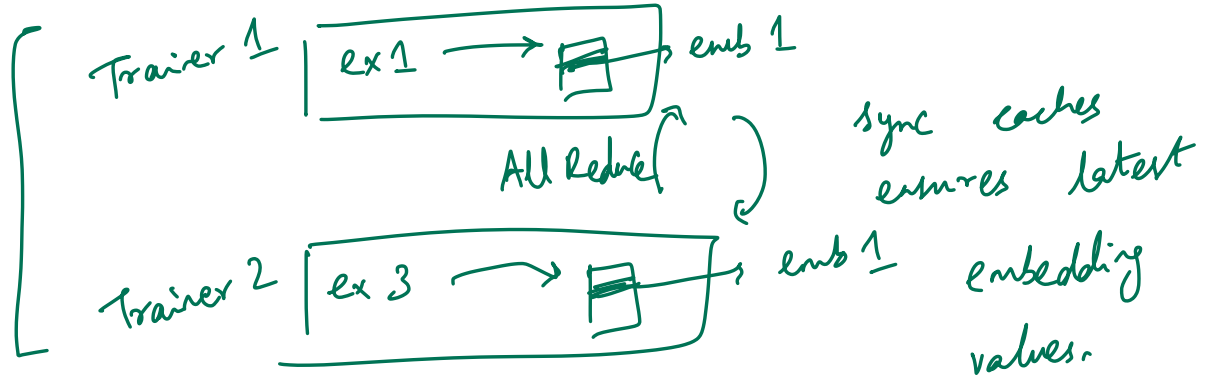
At the end of each iteration, each trainer synchronizes caches



Cache

global batch

why do you need to do this?



SUMMARY

Recommendation models: Embeddings access overheads

BagPipe: Efficient distributed training

- Lookahead to pre-fetch and cache embeddings

- Cache synchronization across trainers



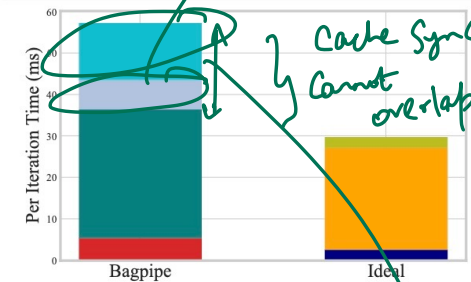
DISCUSSION

<https://forms.gle/9wKQVqNNWcGLJd7N9>

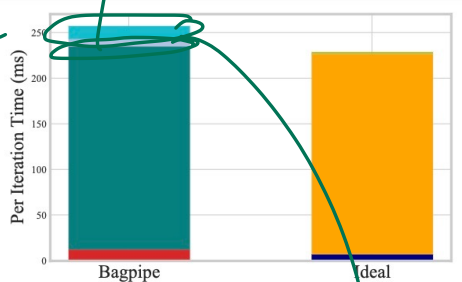
Consider a recommendation model trained on a graph where we use 2-hop neighbors. What are some challenges in using BagPipe-style ideas for such a workload?

Legend: Forward-Bagpipe (red), Forward-Ideal (dark blue), Backward+MLPsync-Bagpipe (teal), Backward+MLPsync-Ideal (orange), Get-Embedding Bagpipe (light blue), Get-Embedding Ideal (yellow-green), Cache Synchronization (cyan)

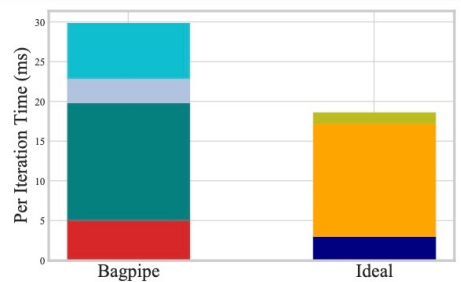
Handwritten notes: "overlap of Get Emb is not perfect" (with arrow pointing to the top of the Bagpipe bars), "Cache Sync cannot overlap" (with arrow pointing to the overlap between Forward and Backward in Bagpipe bars), "Cache Synchronization" (with arrow pointing to the cyan segment in the Ideal bars).



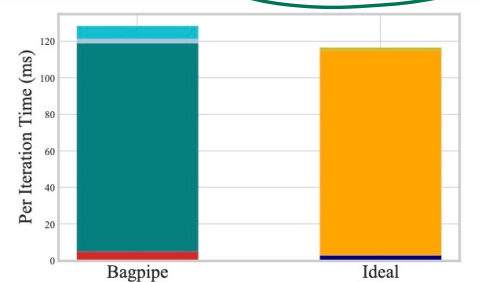
(a) DLRM: p3.2xlarge



(b) DeepFM: p3.2xlarge



(c) DLRM: g5.8xlarge



(d) DeepFM: g5.8xlarge

Handwritten note: "smaller model" (with arrow pointing from (a) to (c)).

Handwritten note: "Cache Synchronization" (with arrow pointing to the cyan segment in the Ideal bars).

NEXT STEPS

Next class: Serverless computing