Pytorch-BigGraph: A Large-scale Graph Embedding System

Facebook AI Research

SysML 2019

Presenter: Zhipeng Zhang

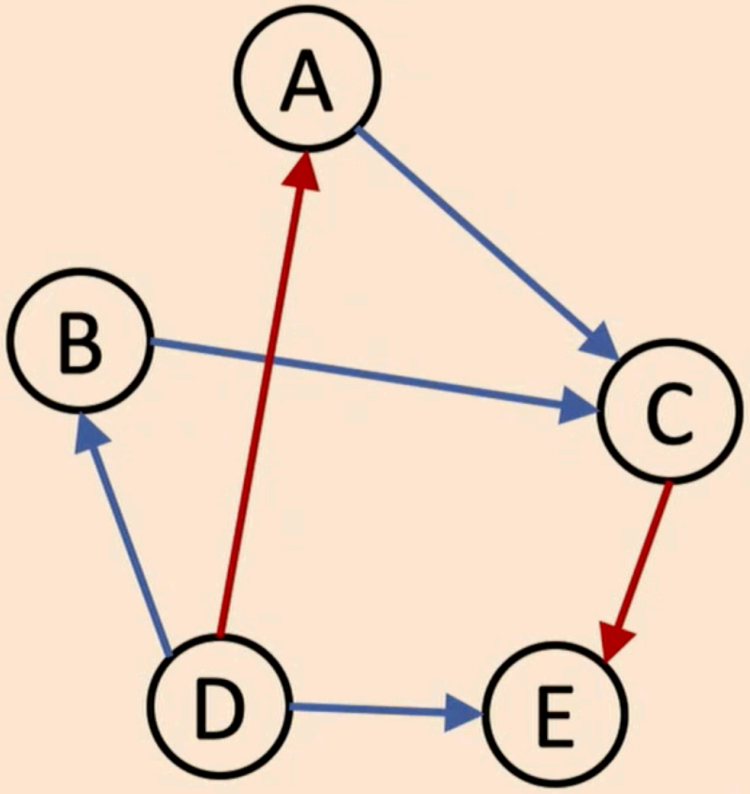
[Adapted from Adam Lerer]

Target Problem

- Knowledge graph embedding.
  - no graph neural networks (e.g., each node has no input features)
  - no random walks

- Platform.
  - Training on single machine, multiple machines (only CPU)

Graph Embeddings

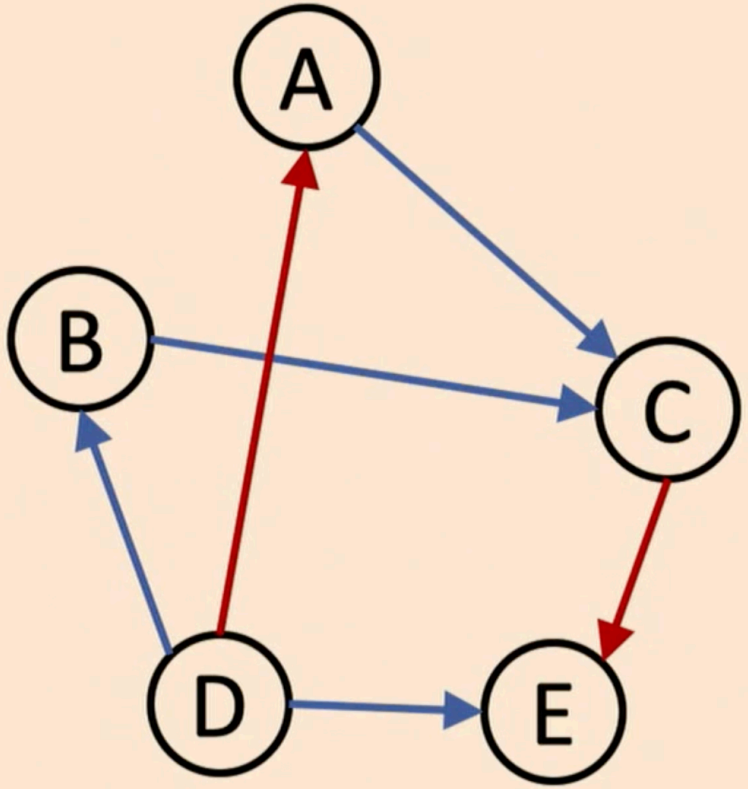


A multi-relation graph

- **Embedding:** A learned map from *entities* to *vectors of numbers* that encodes similarity
  - Word embeddings: word -> vector
  - Graph embeddings: node -> vector

- **Graph Embedding:** Train embeddings with the objective that **connected nodes have more similar embeddings** than unconnected nodes
  - Not the same as graph neural networks. GNNs are a parametric, supervised model over graphs.

Why Graph Embeddings?

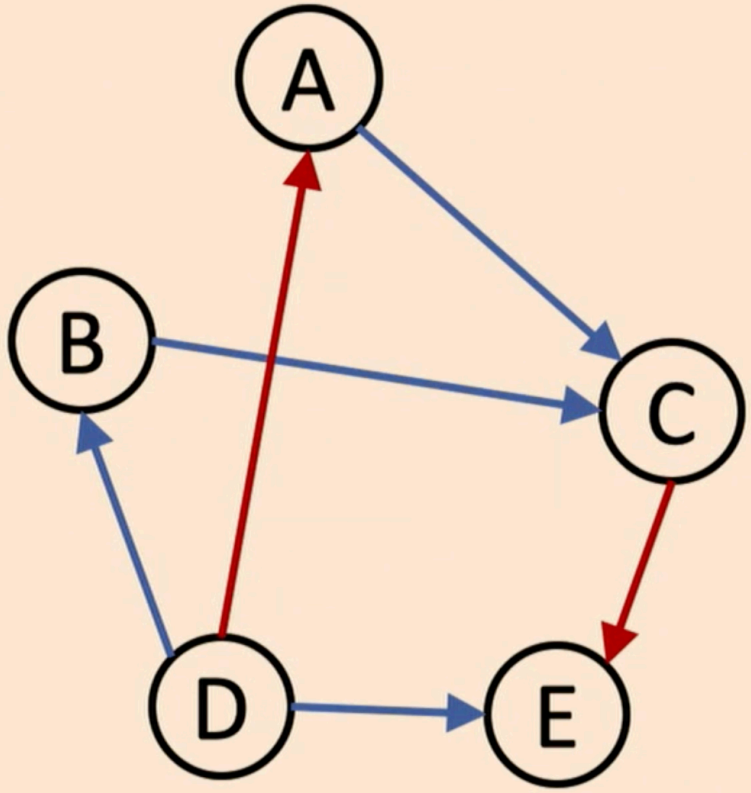


A multi-relation graph

Like word embeddings, graph embeddings are a form of **unsupervised learning** on graphs.

- **Task-agnostic** node representations
- Features are useful on downstream tasks without much data
- Nearest neighbors are semantically meaningful

Graph Embeddings



A multi-relation graph

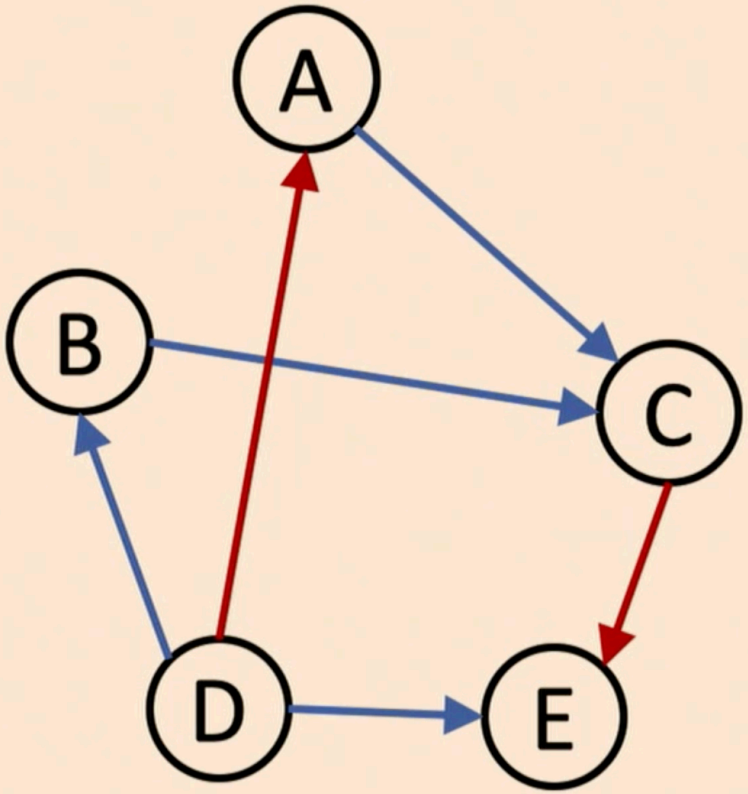
Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

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

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. **"Complex embeddings for simple link prediction."** *ICML, 2016*

Graph Embeddings



A multi-relation graph

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

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

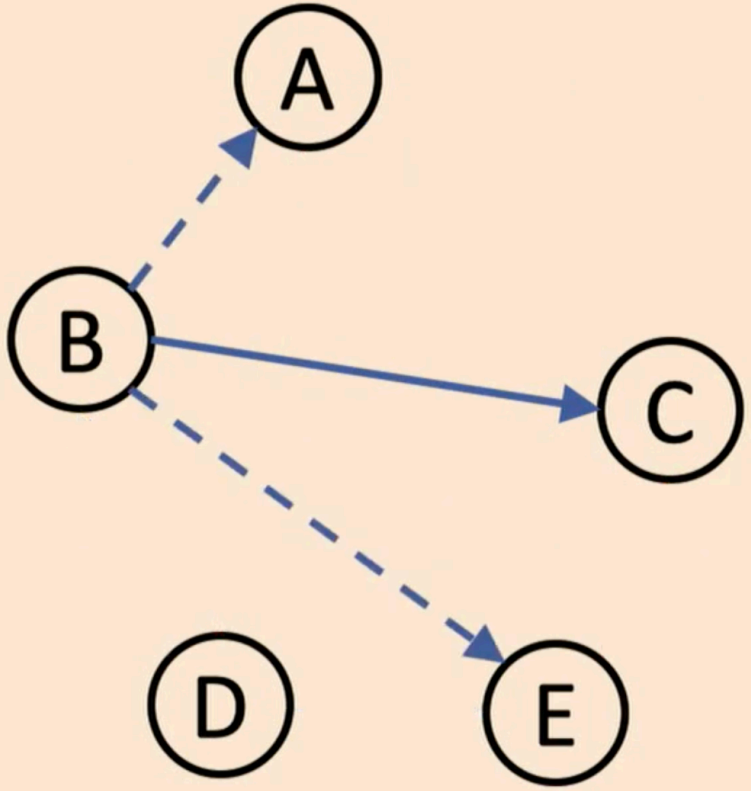
The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. **"Complex embeddings for simple link prediction."** *ICML, 2016*

Graph Embeddings



A multi-relation graph

Margin loss between the score for an edge $f(e)$ and a negative sampled edge $f(e')$

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

The score for an edge is a similarity (e.g. dot product) between the source embedding and a transformed version of the destination embedding, e.g.

$$f(e) = \cos(\theta_s, \theta_r + \theta_d)$$

Negative samples are constructed by taking a real edge and replacing the source or destination with a random node.

$$S'_e = \{(s', r, d) | s' \in V\} \cup \{(s, r, d' | d' \in V\}$$

Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., and Yakhnenko, O. **"Translating embeddings for modeling multi-relational data."** *NIPS, 2013*

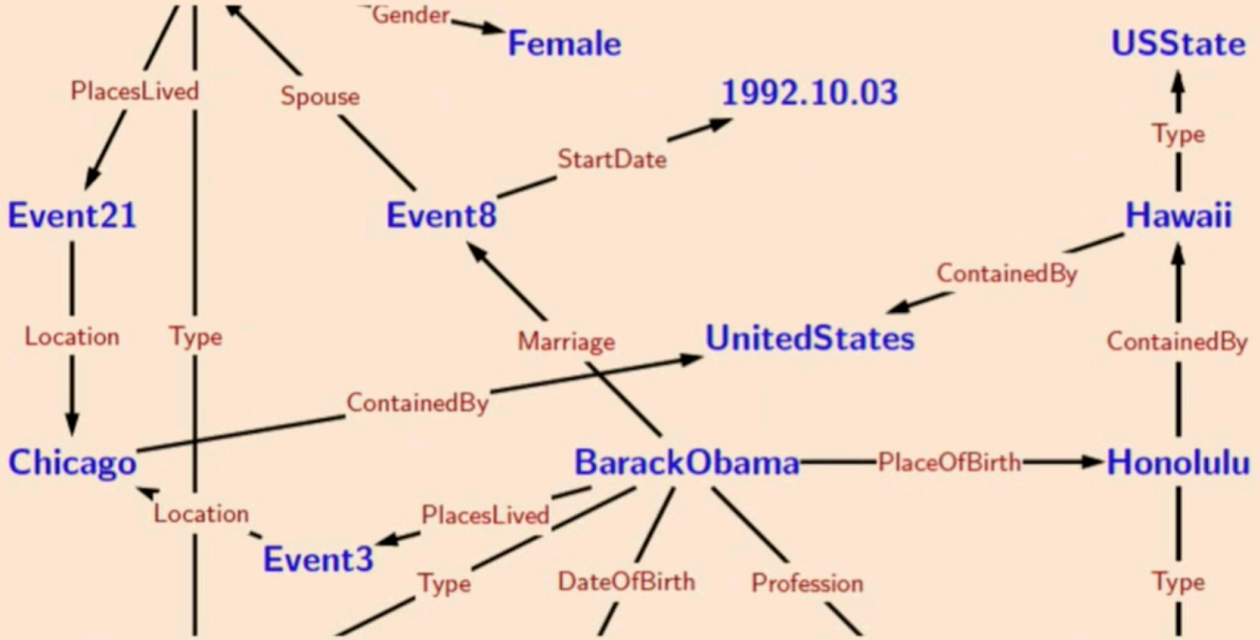
Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. **"Complex embeddings for simple link prediction."** *ICML, 2016*

Commonly-Studied Datasets

Knowledge Graphs

Standard domain for studying graph embeddings

(Freebase, …)



Recommender Systems

Deals with graph-like data, but supervised

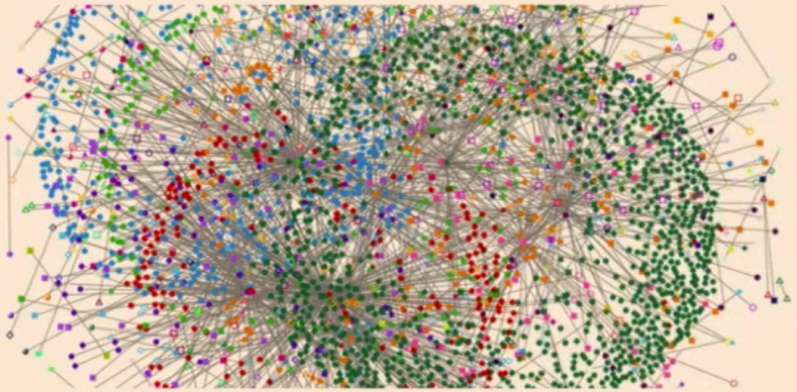
(MovieLens, …)

	user_id	movie_id	rating
0	196	242	3
1	186	302	3

Social Graphs

Predict attributes based on homophily or structural similarity

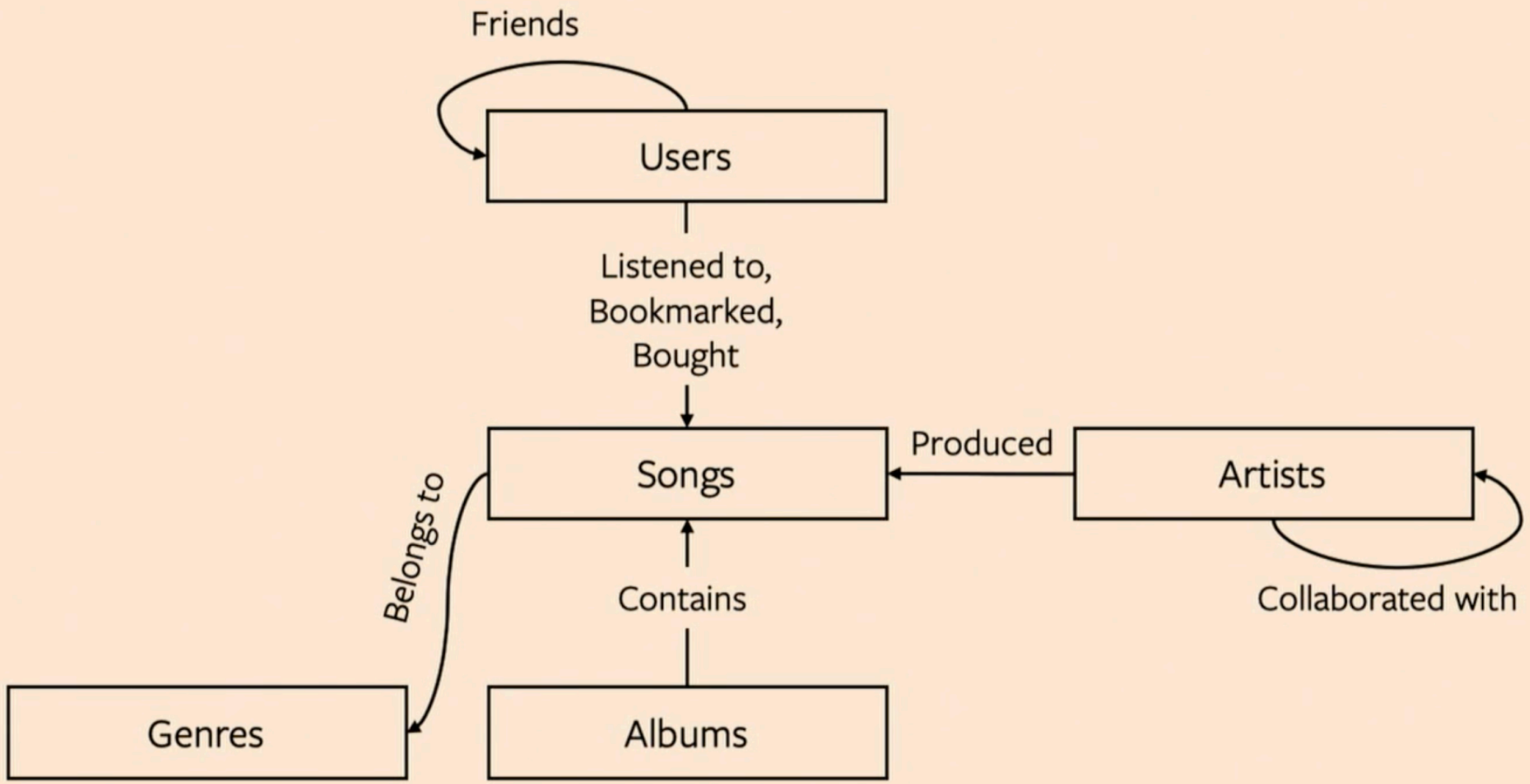
(Twitter, Yelp, …)



Wang, Zhenghao & Yan, Shengquan & Wang, Huaming & Huang, Xuedong. (2014). An Overview of Microsoft Deep QA System on Stanford WebQuestions Benchmark.

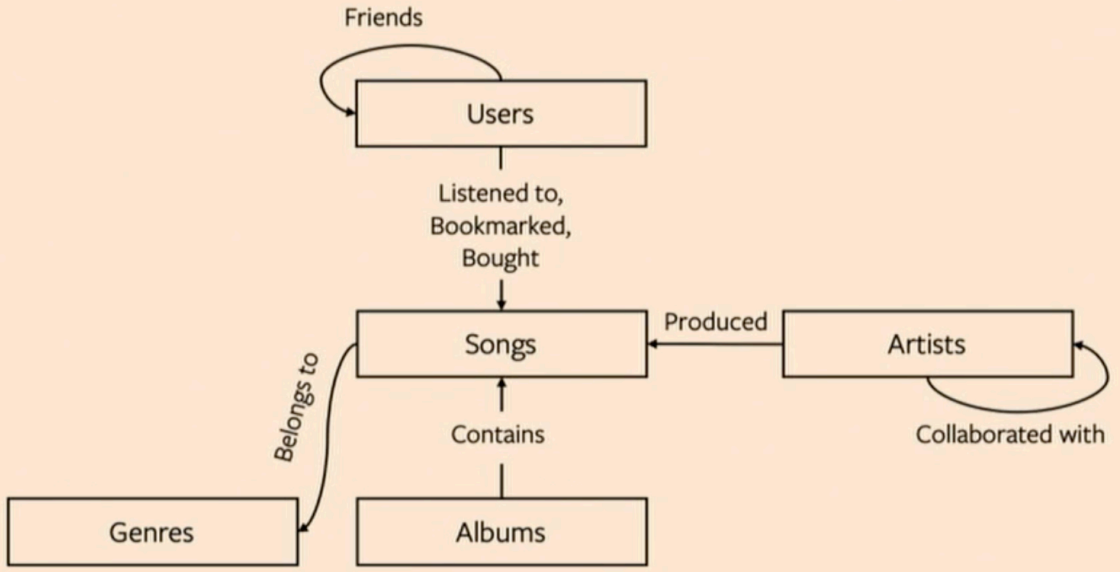
https://threatpost.com/researchers-graph-social-networks-spot-spammers-061711/75346/

(Big) Graph Data is Everywhere

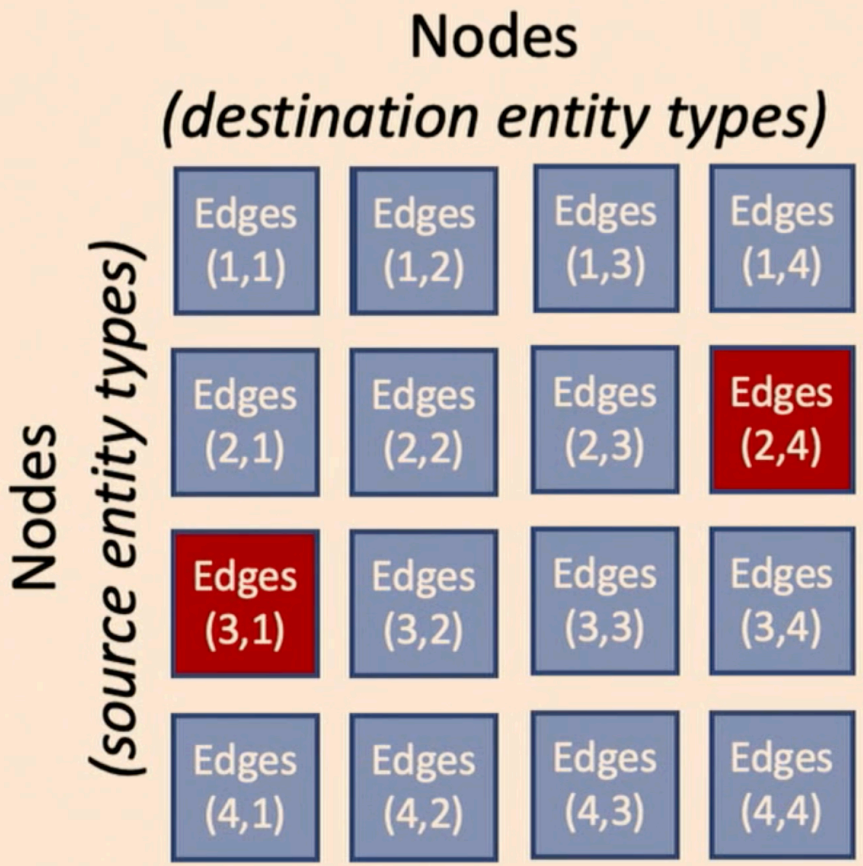


Online Interaction Datasets Are Big Data

- Web companies have graphs with billions of nodes, trillions of edges
- 2B nodes x 100d = **800 GB model**
  - Doesn't fit in (CPU) memory
- Training speed is also an issue for graphs of this size

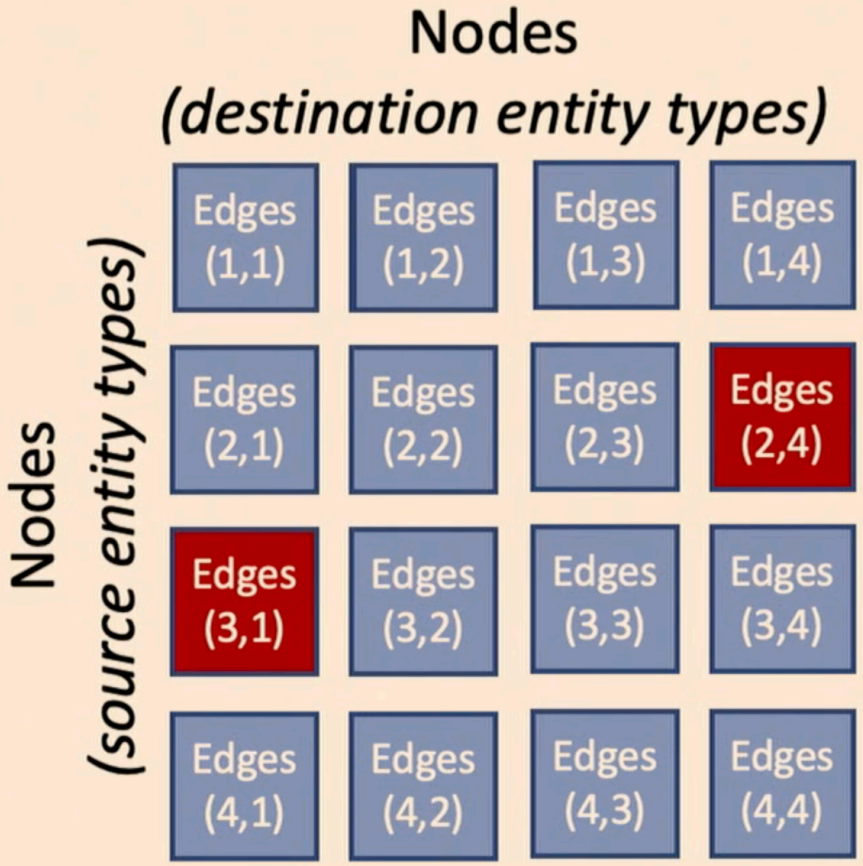


Node Partitioning



- Use **matrix blocking** to subdivide the graph
- Nodes are divided uniformly into N shards
- Edges divided into N^2 buckets by source and destination node shard
  - (relation embeddings are not sharded)

Node Partitioning

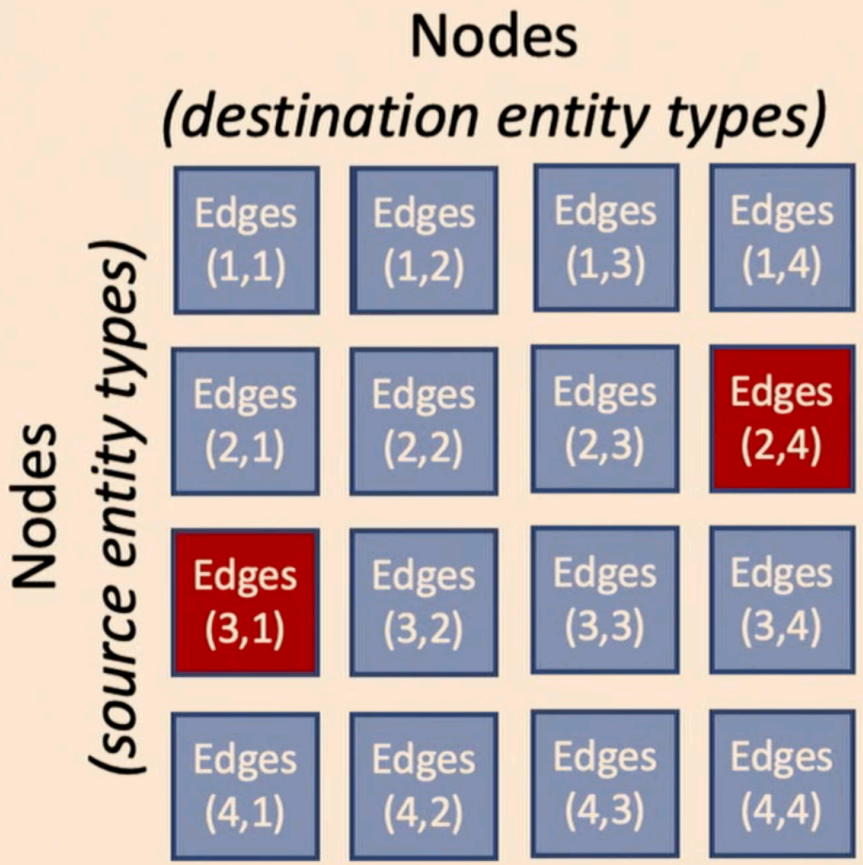


Single machine: only 2 partitions of the model need to be held in memory. Others swapped to disk.

- Multi-threaded on CPU; no GPU required

Distributed training: Buckets with disjoint partitions can be trained simultaneously without synchronizing model parameters.

- Communication is only required when switching buckets



Effects on training objective:

- Edges are not sampled i.i.d., grouped by bucket
  - Theoretically justified; converges to the minimum of the true loss (Gemulla et al., 2011)
- Distribution of negative samples changes
  - Less relevant for larger graphs
  - Can be ameliorated by shuffling partitions periodically

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. **"Large-scale matrix factorization with distributed stochastic gradient descent."** *KDD 2011*

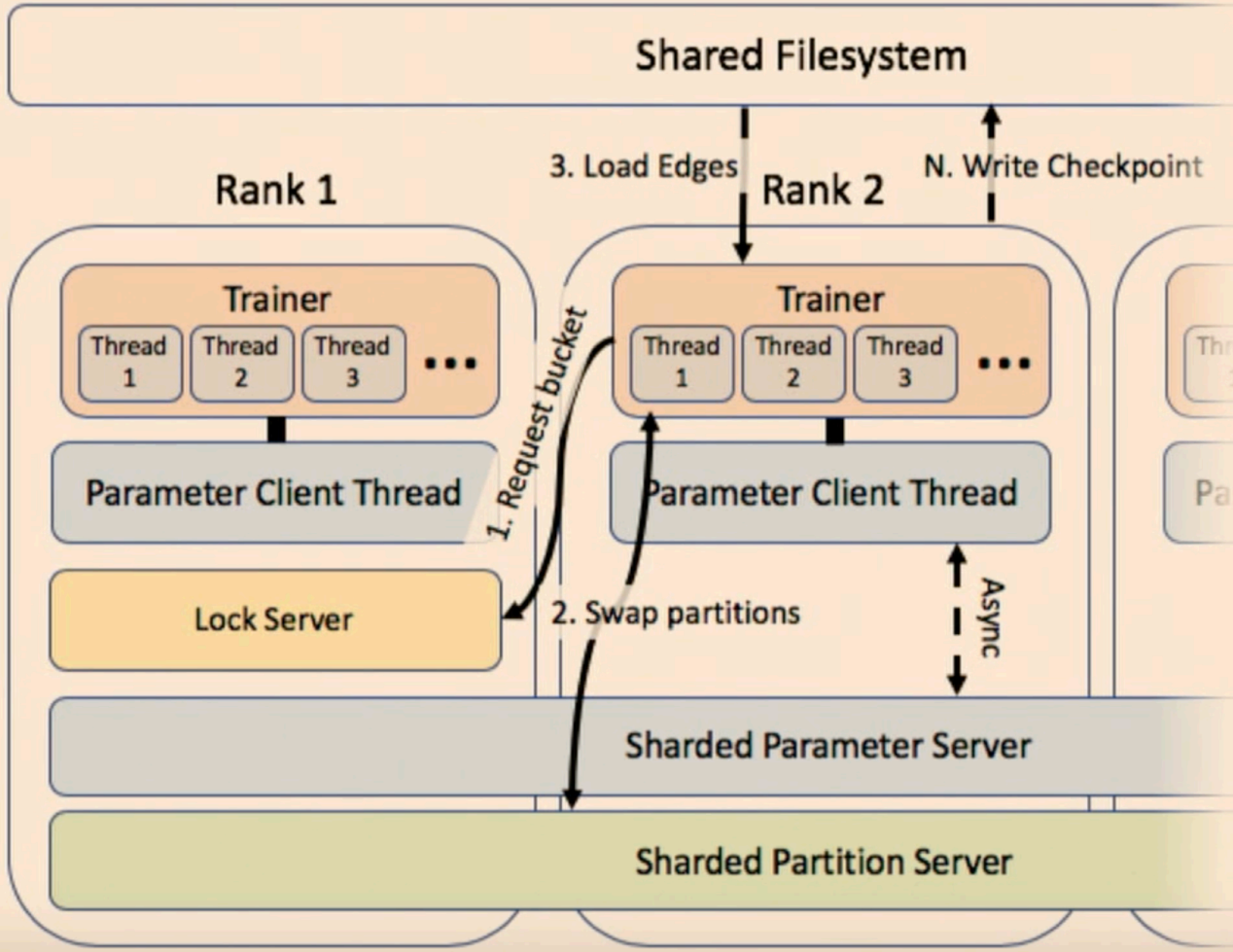
Distributed Training

Distributed training requires several types of communication.

Synchronizing bucket access

Exchanging model partitions

Sharing common parameters



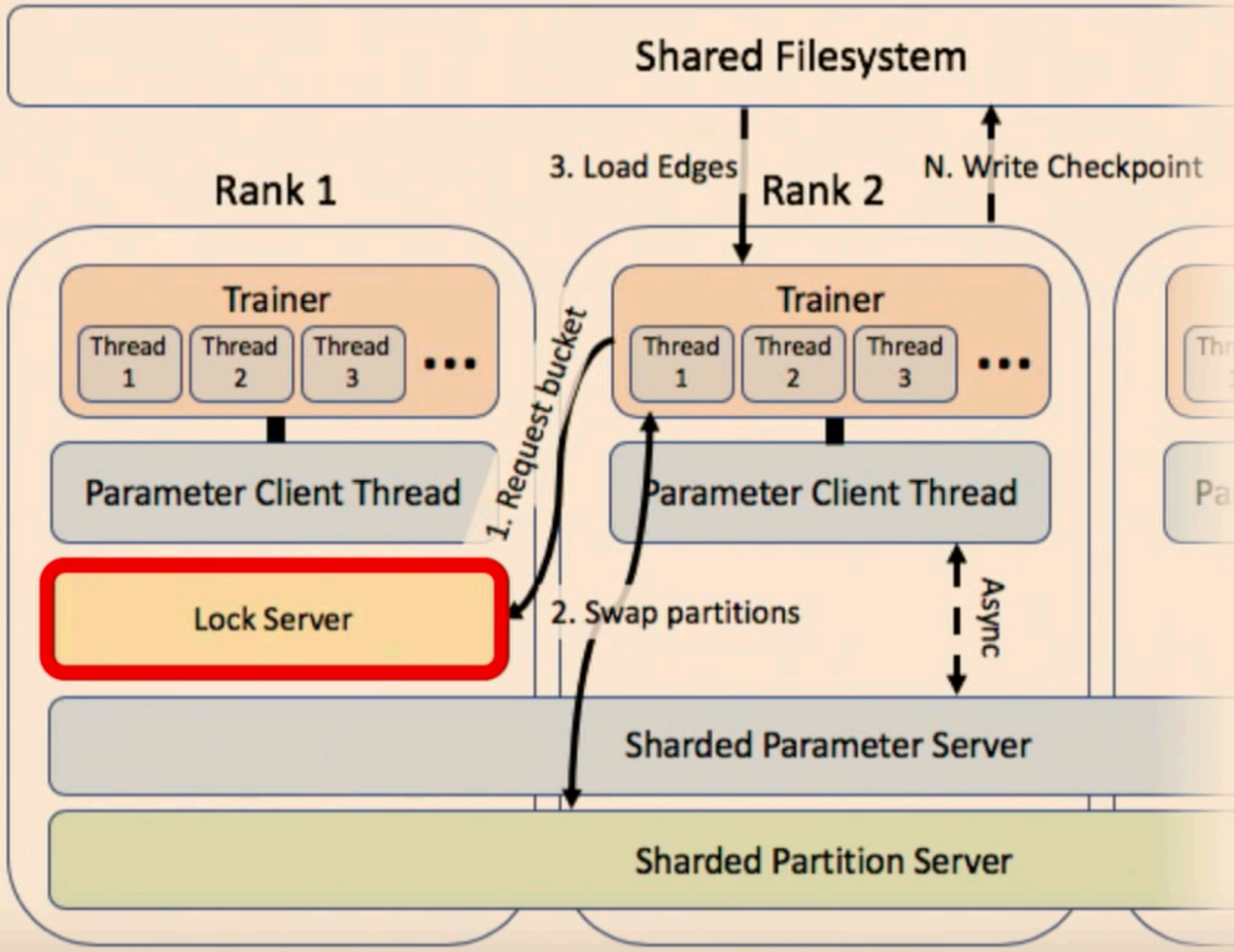
Distributed Training

Synchronizing bucket access

- *Lock server* assigns buckets to machines
- Greedily maximizes machine-partition affinity

Exchanging model partitions

Sharing common parameters



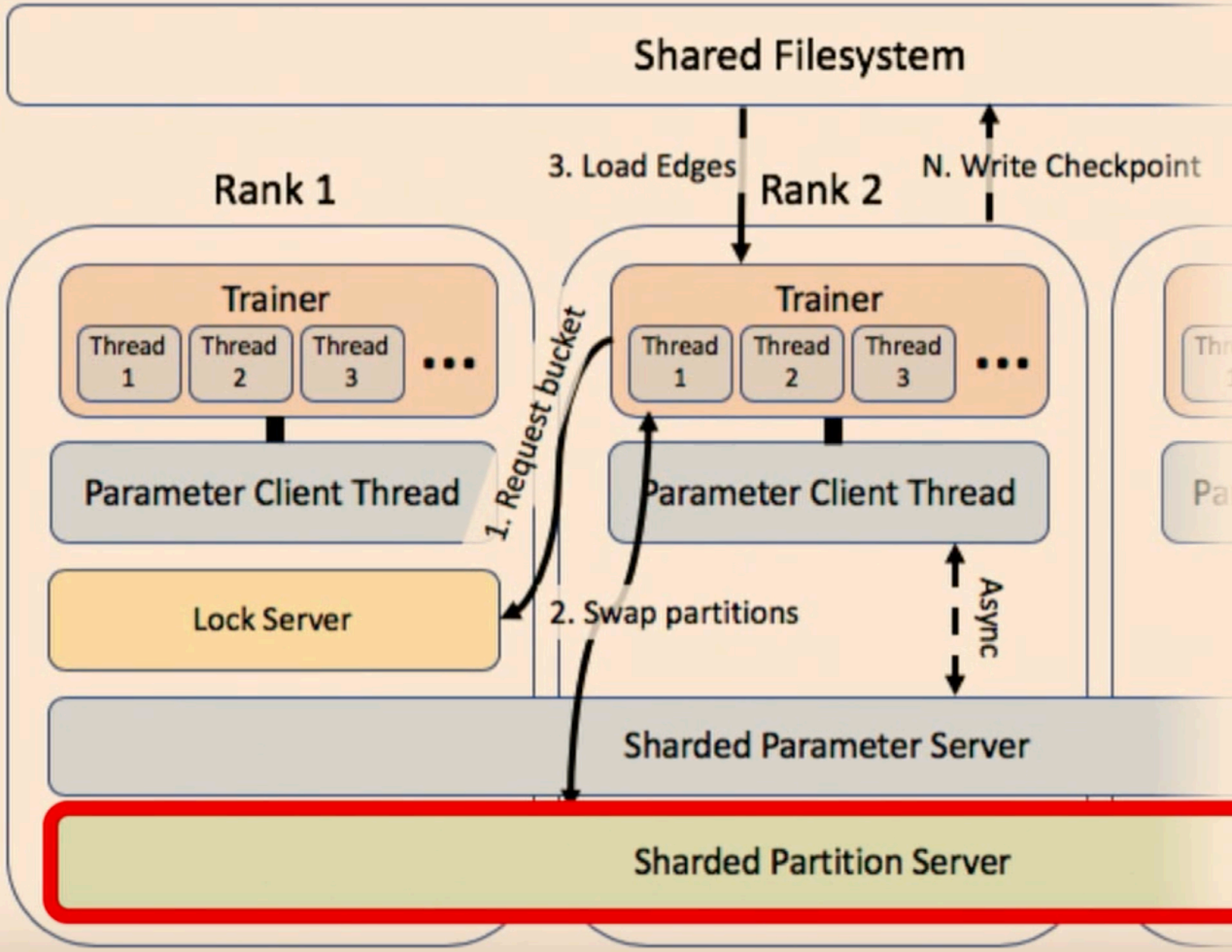
Distributed Training

Synchronizing bucket access

Exchanging model partitions

- When not in use, model partitions are stored on a sharded *partition server* running on the training machines
- Alternatively, partitions can be swapped to disk to save memory

Sharing common parameters



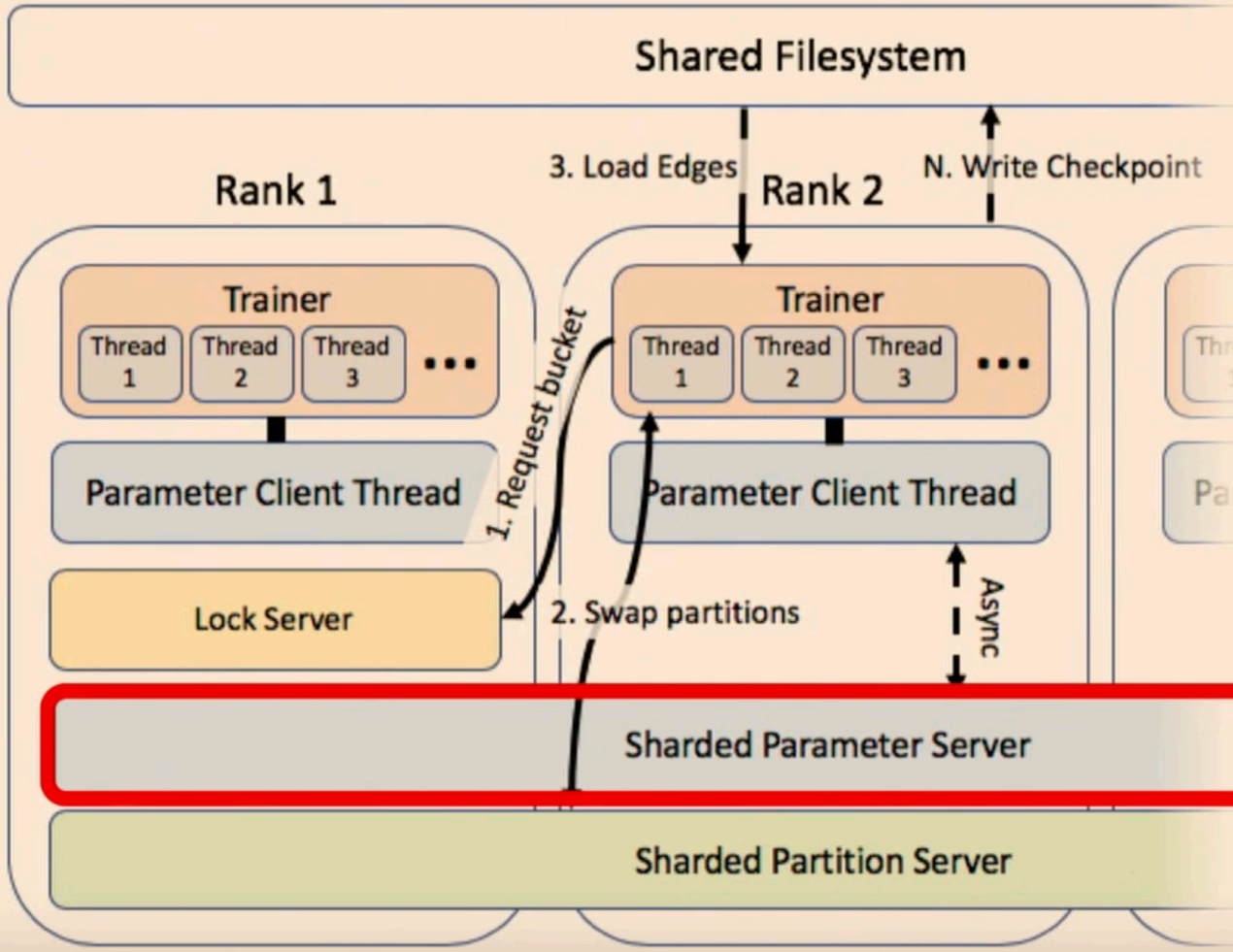
Distributed Training

Synchronizing bucket access

Exchanging model partitions

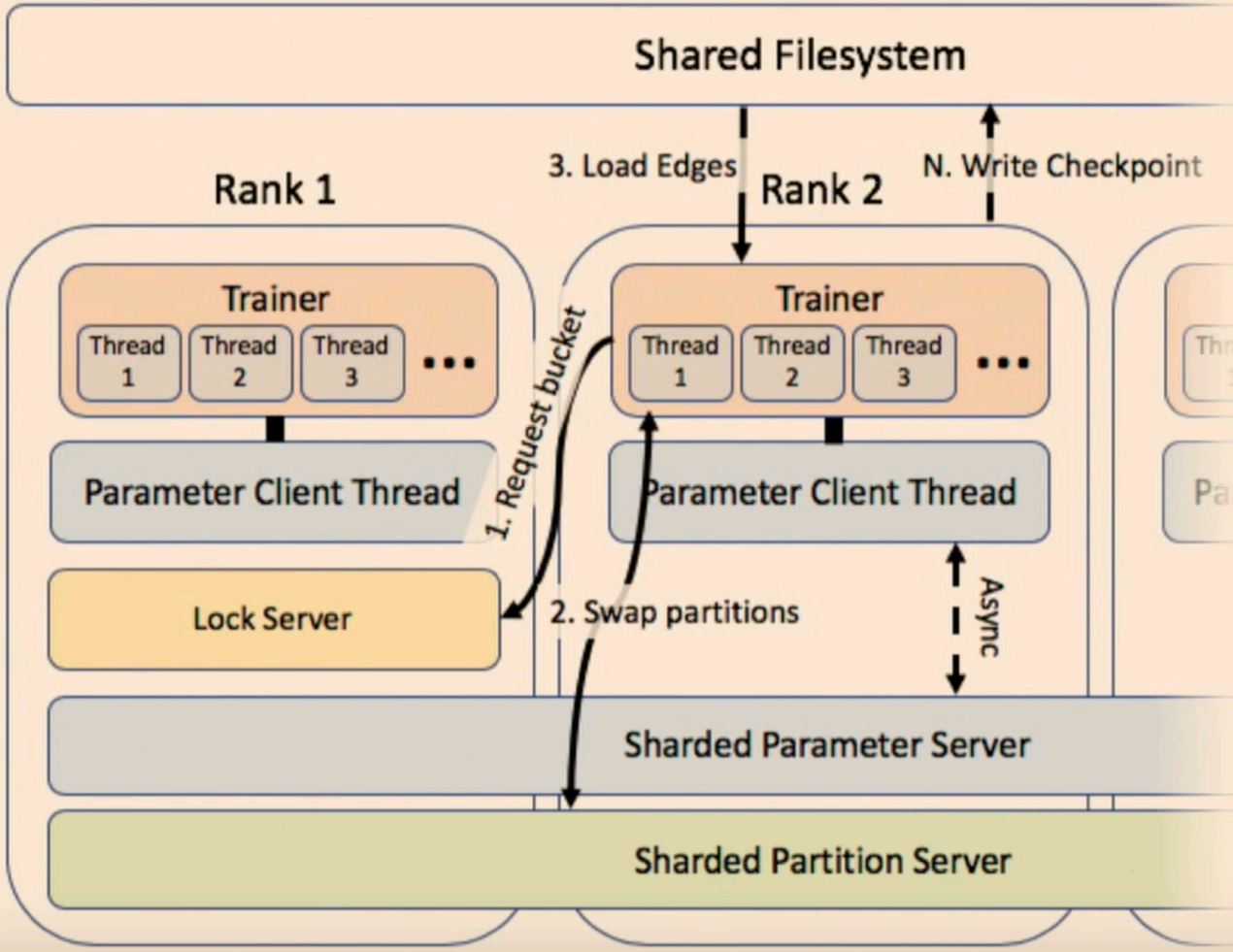
Sharing common parameters

- A small number of model parameters cannot be partitioned, e.g. relation operators
- These parameters are shared across machines asynchronously via a standard parameter server sharded across all machines
- Write gradients, read parameters



Distributed Training

- Distributed training implemented with MPI-like PyTorch distributed primitives (*torch.distributed*)
- Runs on MPI, TCP, or gloo.



Optimized Negative Sampling

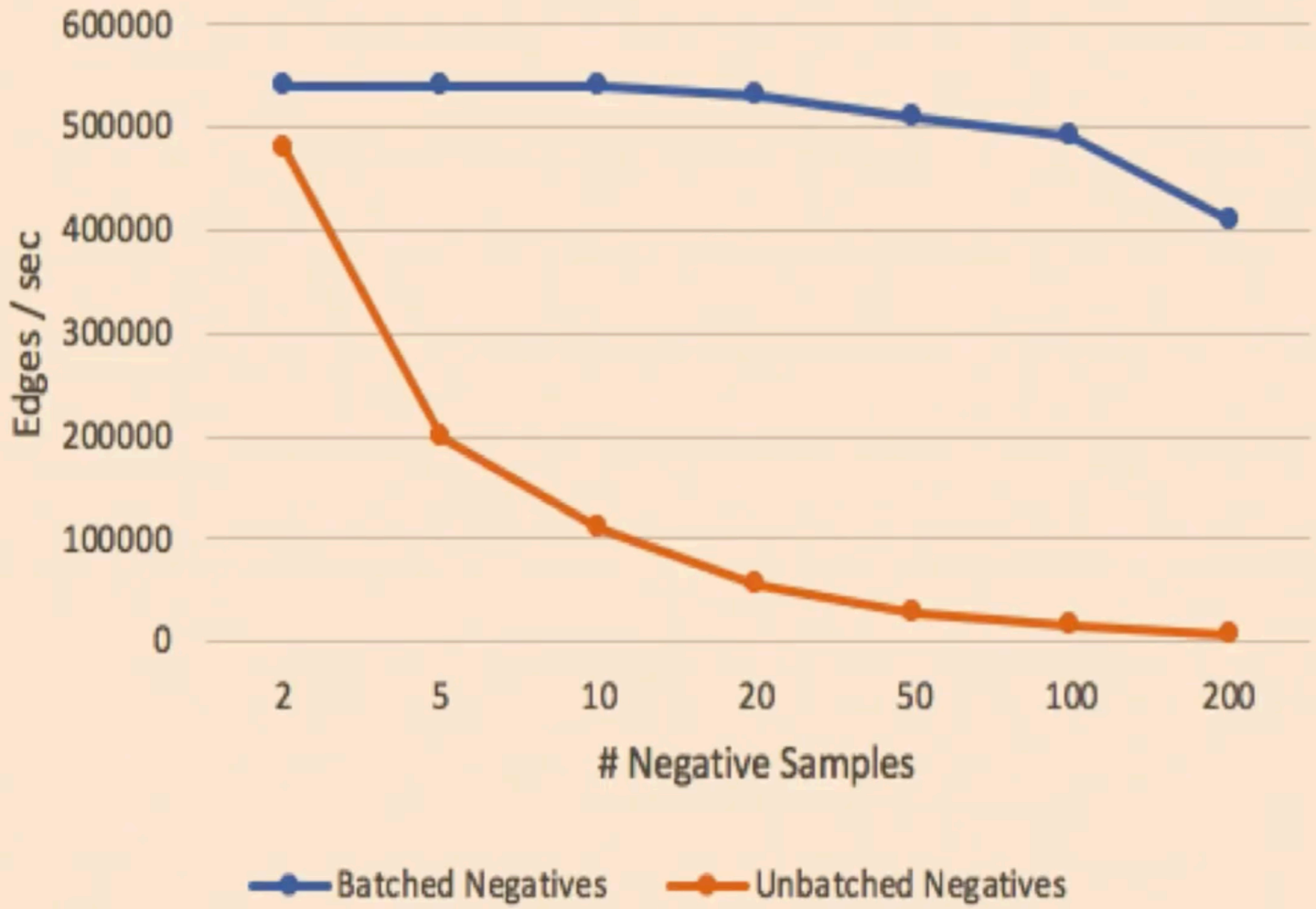
- Training time dominated by negative sampling edges
  - Negative samples constructed by replacing an edge's source or destination node with a "random" node
  - Typically >>1 negative sample per edge

Optimized Negative Sampling

- Training time dominated by negative sampling edges
  - Negative samples constructed by replacing an edge's source or destination node with a "random" node
  - Typically >>1 negative sample per edge
- What if we corrupt a sub-batch of 100 edges with the same set of random nodes?
  - Reduces random-access memory bandwidth by a factor of 100
  - If the similarity metric is dot product or cosine distance, a batch of negative scores S can be computed as a matrix multiply

$$f(e') = \theta_{s,pos}\theta_{d,neg}^{T}$$

Optimized Negative Sampling



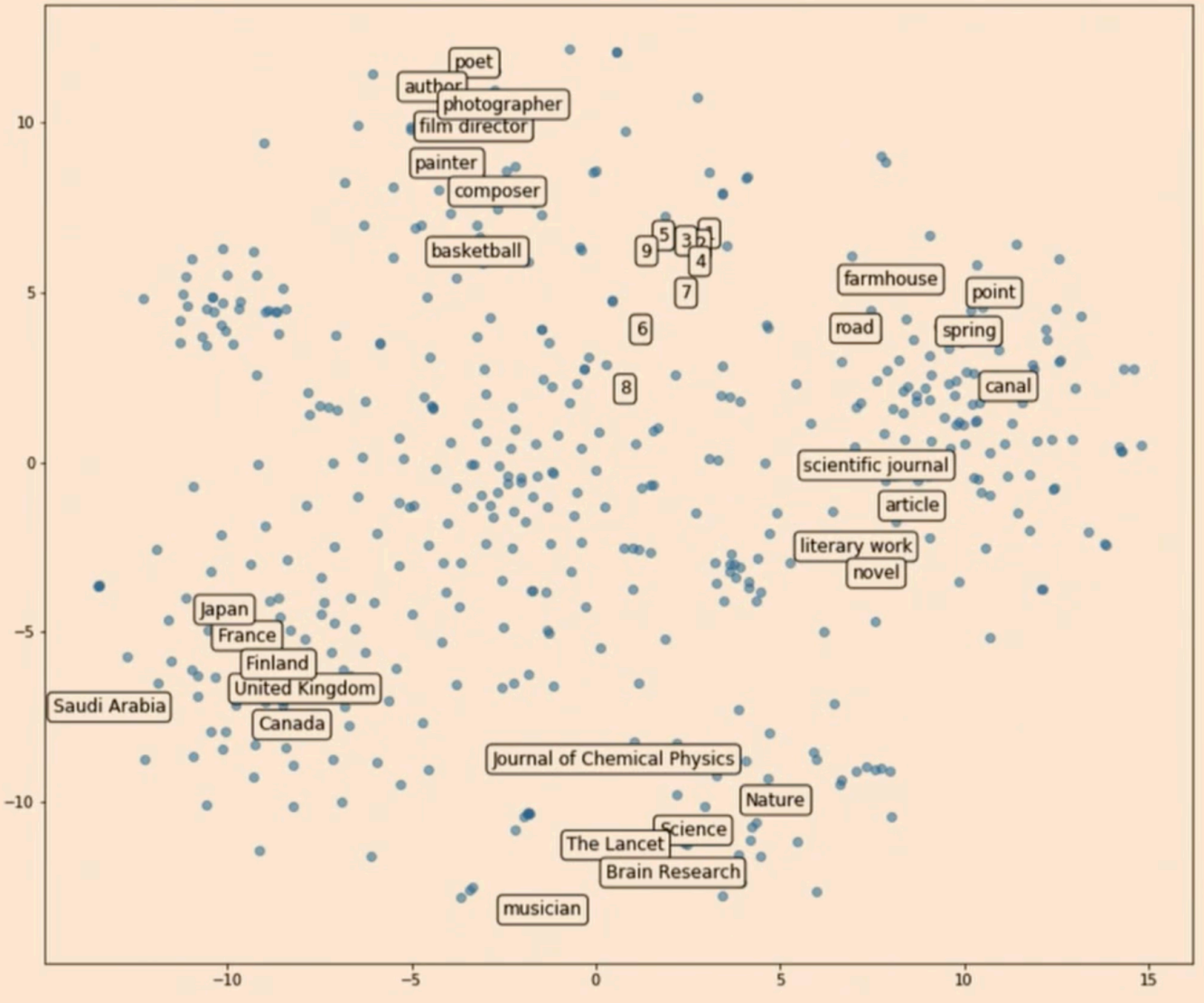
Freebase Knowledge Base Completion (FB15k Subset)

Method	MRR		Hit@10
	Raw	Filtered	
RESCAL (Nickel et al., 2011)	0.189	0.354	0.587
TransE (Bordes et al., 2013)	0.222	0.463	0.749
HolE (Nickel et al., 2016b)	0.232	0.524	0.739
ComplEx (Trouillon et al., 2016)	0.242	0.692	0.840
R-GCN+ (Schlichtkrull et al., 2018)	0.262	0.696	0.842
StarSpace (Wu et al., 2018)	-	-	0.838
Reciprocal ComplEx-N3 (Lacroix et al., 2018)	-	0.860	0.910
PBG (TransE)	0.265	0.594	0.785
PBG (ComplEx)	0.242	0.790	0.872

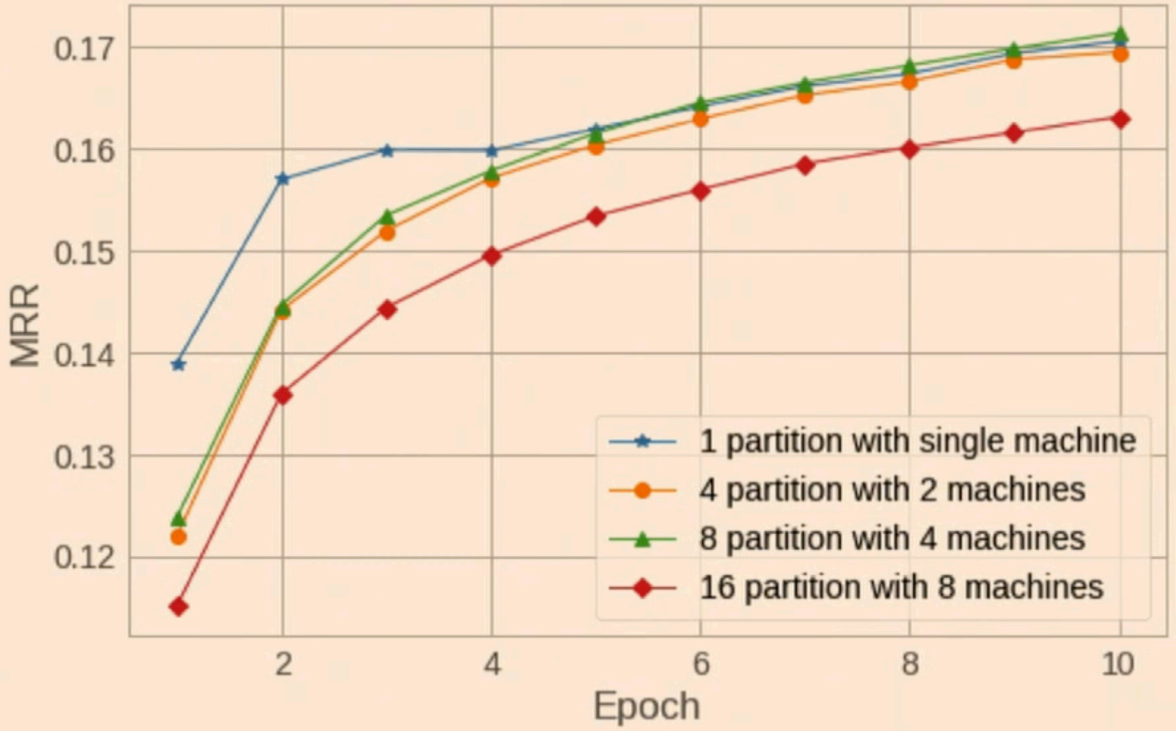
Embedding the Full Freebase Knowledge Graph

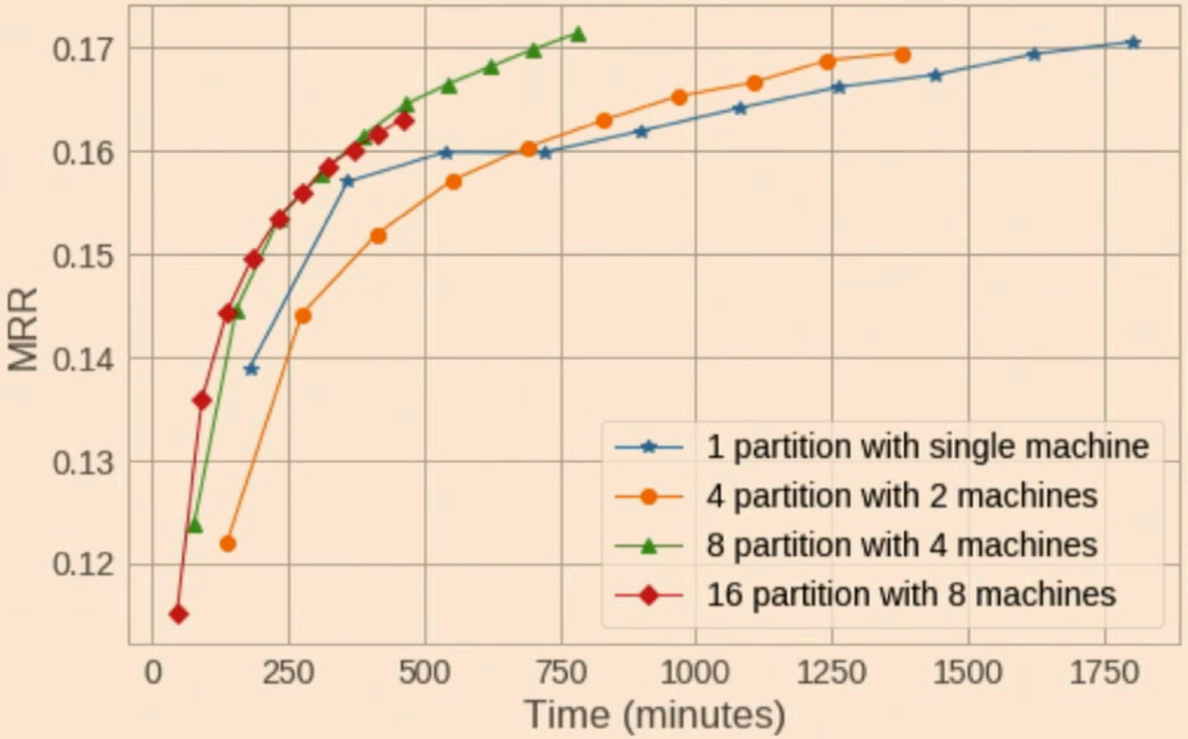
- We then train embeddings on the full Freebase graph
  - 51M nodes, 1B edges
- More partitions → less memory

# Parts	MRR	Mem (GB)
1	0.170	59.6
4	0.174	30.4
8	0.172	15.5
16	0.174	6.8

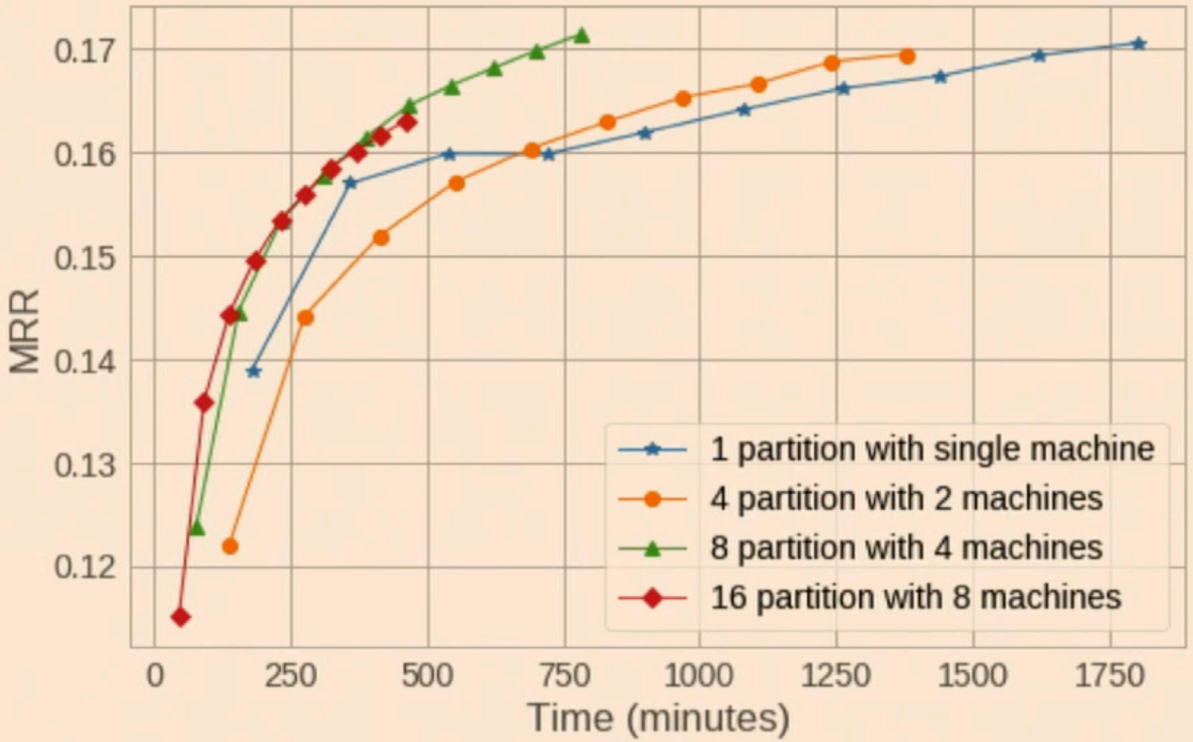


Freebase Parallel Training

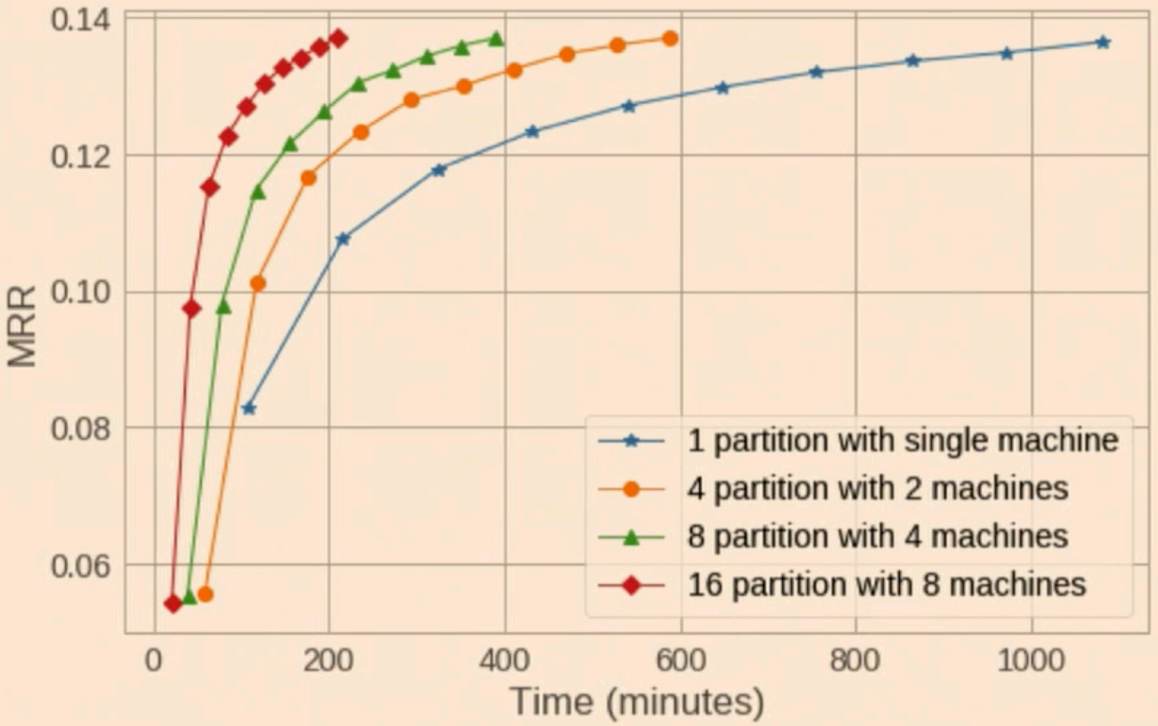




Freebase



Twitter



Knowledge graphs are finicky to train, but social graphs are not!

Future Work

- Untangling the causes of model degradation, and ameliorating them
- Parallel training for graphs with featurized nodes or node attributes
- Hierarchical matrix blocking for parallel GPU training