

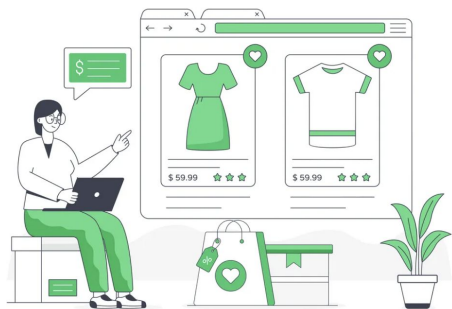
Towards Fully Disaggregated Recommendation Model Serving

Yibo Huang, Yiming Qiu, Zhenning Yang, Yi Dai, Dingming Wu, Fan Lai
Jiarong Xing, Ang Chen



EMR models for recommendation systems are critical today

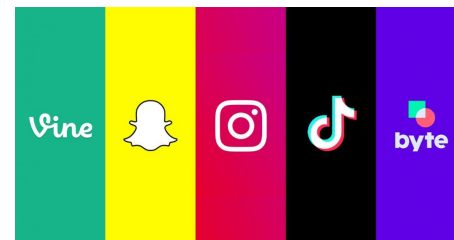
- Embedding-based Recommendation (EMR) models:
 - Efficient sparse data processing, high recommendation accuracy
 - Good personalization for better quality of user experience
 - Widely deployed in production datacenters
- Powering diverse applications:



E-commerce



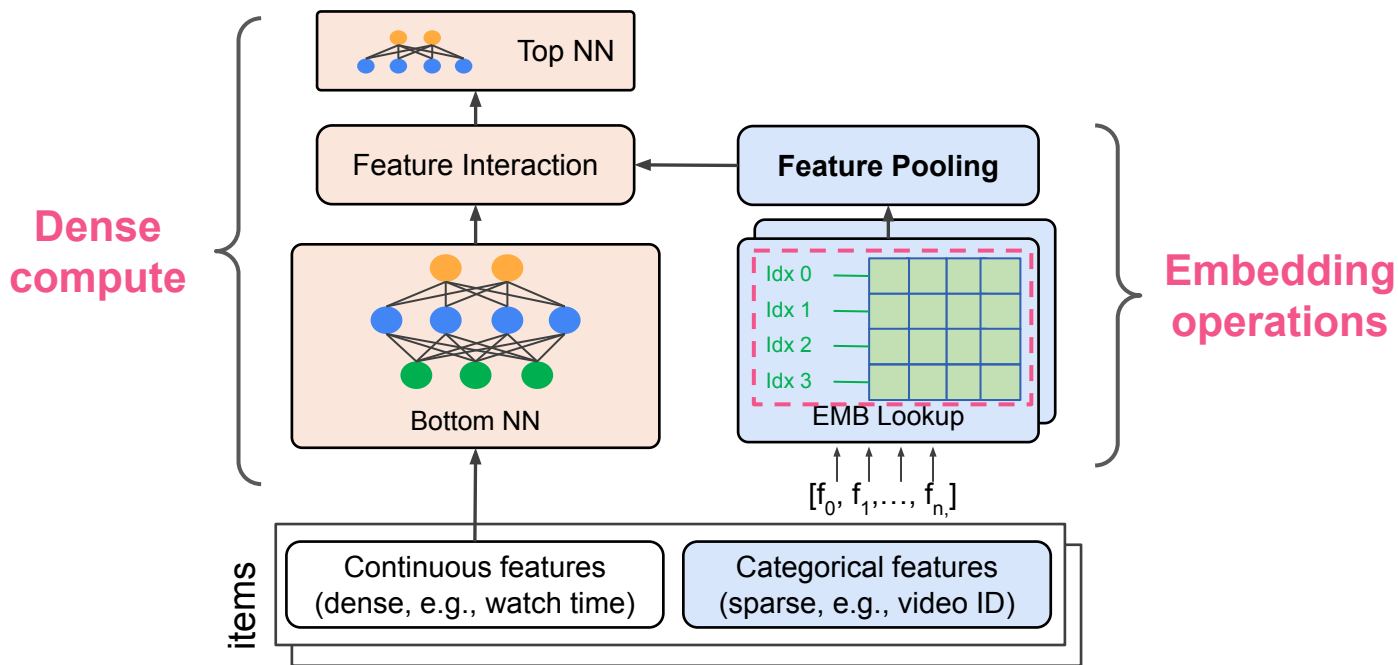
Search engines



Short-video platforms

Background: An example of EMR models – DLRM

- **GPUs** for dense computation
- **CPUs** for embedding lookups/processing
- **DRAM** for embedding store at TB scale



Trend: Variation in the resource requirements of EMR models

- Rapidly evolving EMR models often incorporate updated model architectures

Model
DLRM-RMC1
DLRM-RMC2
DLRM-2020
DLRM-2021
DLRM-2022

Trend: Variation in the resource requirements of EMR models

- Rapidly evolving EMR models often incorporate updated model architectures
 - Increasing size of embedding tables: $O(10)GB \rightarrow O(10)TB$

Model	Memory Demand(GB)
DLRM-RMC1	3.8
DLRM-RMC2	81.10
DLRM-2020	1,236.14
DLRM-2021	2,954.66
DLRM-2022	44,706.42

Trend: Variation in the resource requirements of EMR models

- Rapidly evolving EMR models often incorporate updated model architectures
 - Increasing size of embedding tables: O(10)GB $\rightarrow$ O(10)TB
 - Various and unpredictable CPU/GPU demands

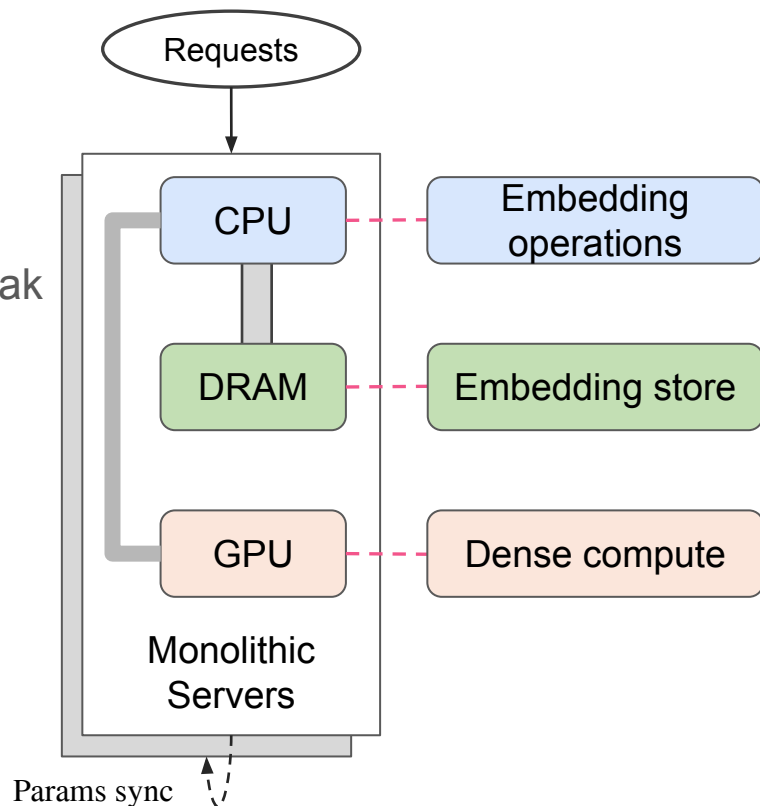
Model	Memory Demand(GB)	CPU GFLOPs	GPU TFLOPs
DLRM-RMC1	3.8	51.2	100.6
DLRM-RMC2	81.10	230.4	50.5
DLRM-2020	1,236.14	6439.08	399.51
DLRM-2021	2,954.66	13949.07	4556.25
DLRM-2022	44,706.42	509.44	235.29

Requirement: Serving EMR models must offer **model upgrade flexibility**, and **independent scaling capability**.

State-of-the-art EMR model serving and limitations

Solution 1: Monolithic machines

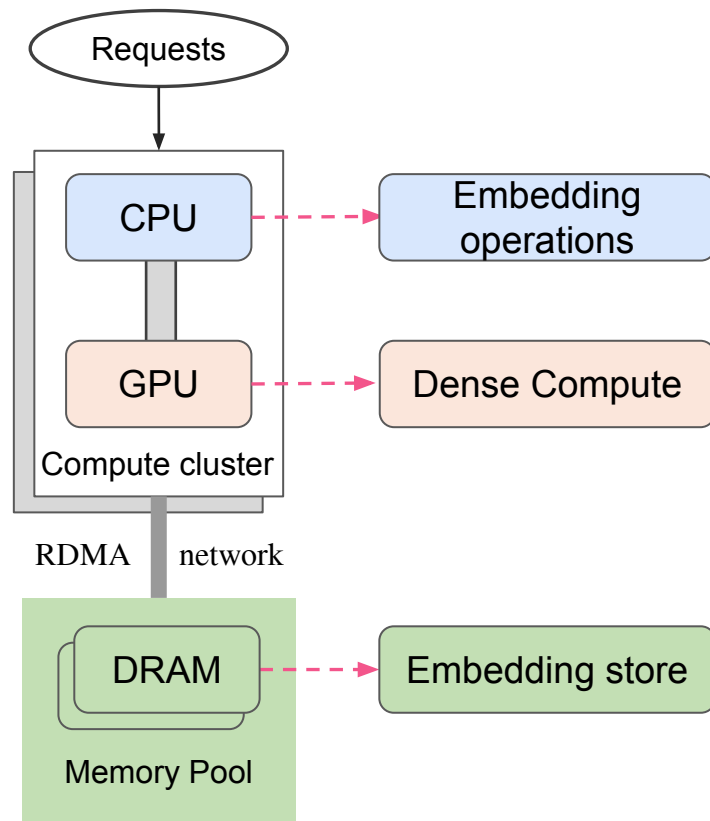
- Fixed resource ratios fail to match actual resource demands of EMR workloads.
- Overprovision GPU, CPU and DRAM based on peak demand of a single resource.
- Wasted costs of up to 23.1% [1].



State-of-the-art EMR model serving and limitations

Solution 2: Partially disaggregated approaches

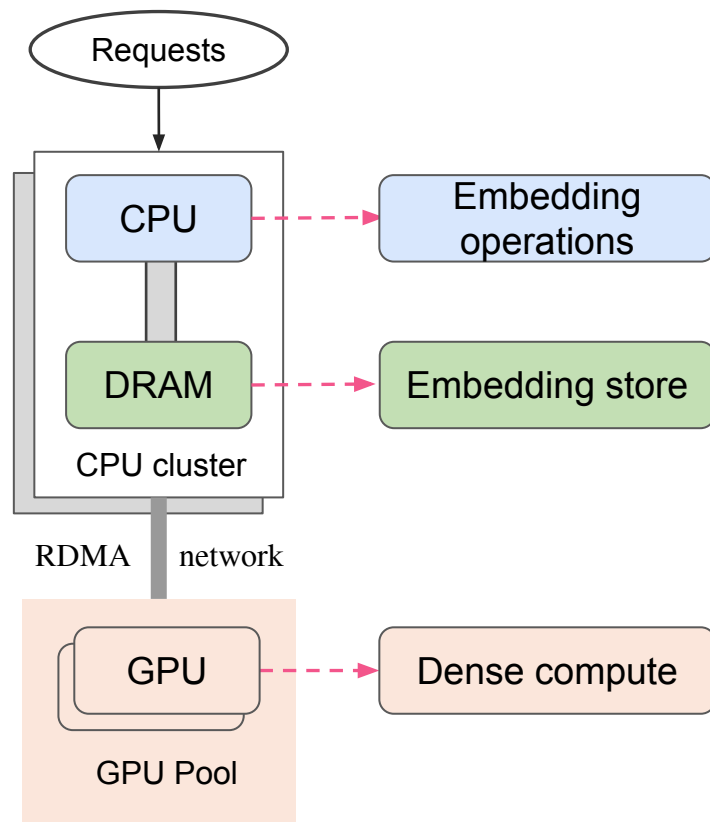
- 2a- Memory-disaggregated:
 - Leaving CPUs and GPUs tightly coupled



State-of-the-art EMR model serving and limitations

Solution 2: Partially disaggregated approaches

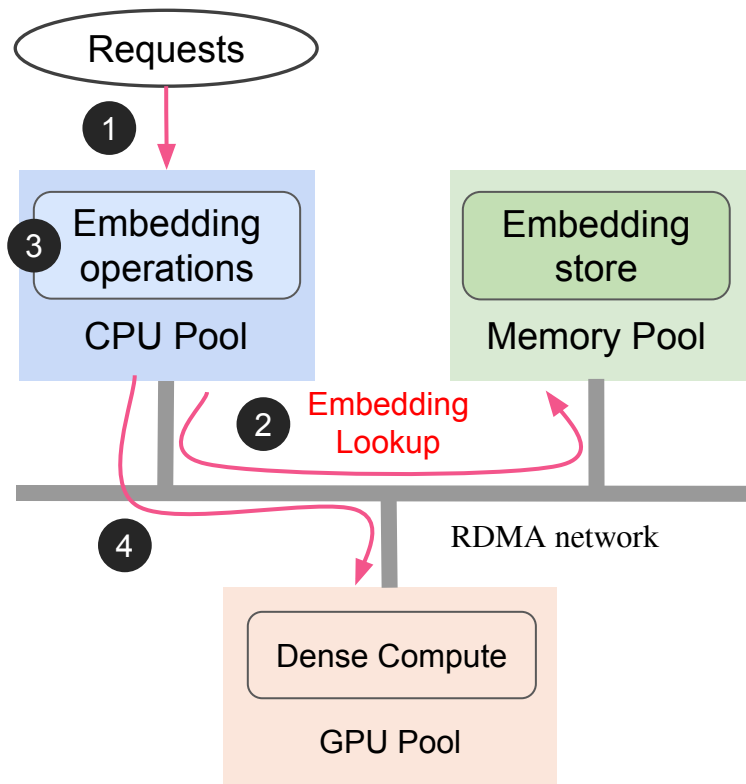
- 2a- Memory-disaggregated:
 - Leaving CPUs and GPUs tightly coupled
 - 2b- GPU-Disaggregated:
 - CPUs and DRAMs are provisioned together
-
- Unable to scale components independently
 - Suboptimal resource efficiency and costs



FlexEMR Vision: Fully disaggregated recommendation model serving

Promise:

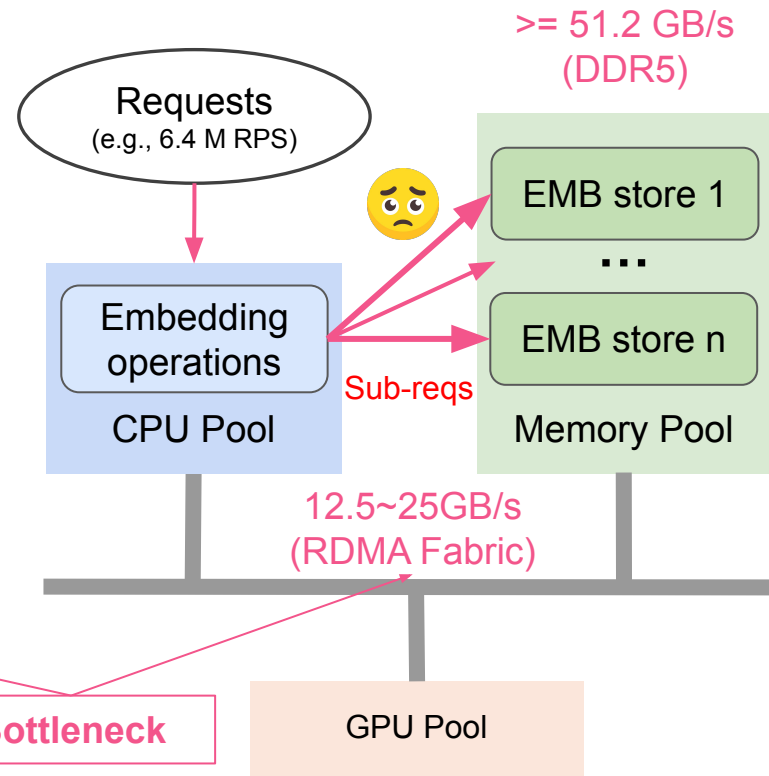
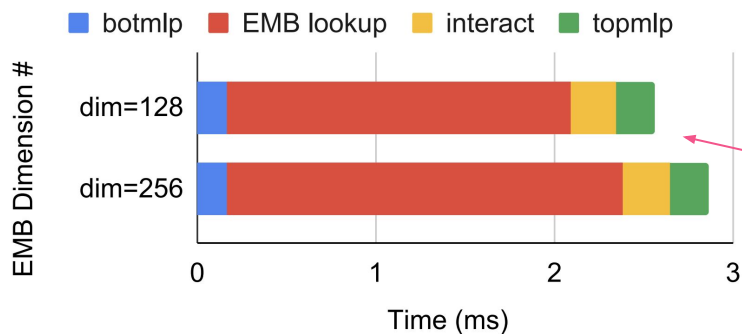
- Flexible scalability
 - Independent scaling
- Cost efficiency
 - Provision on demand
 - Reduce total cost of ownership (TCO)
- Improved robustness
 - Fault isolation



Challenge 1: Large-scale fan-out pattern

Remote embedding lookup:

- One request has ~100 feature indices (i.e., subrequests).
- One feature index $\rightarrow$ one embedding vector with ~1K bytes.
- Tens of embedding (EMB) stores (e.g., partitions/shards).
- One batch contains ~1000 lookups.



Design 1: Locality-aware embedding lookup

Key insight: Leverage embedding locality to reduce network traffic contention.

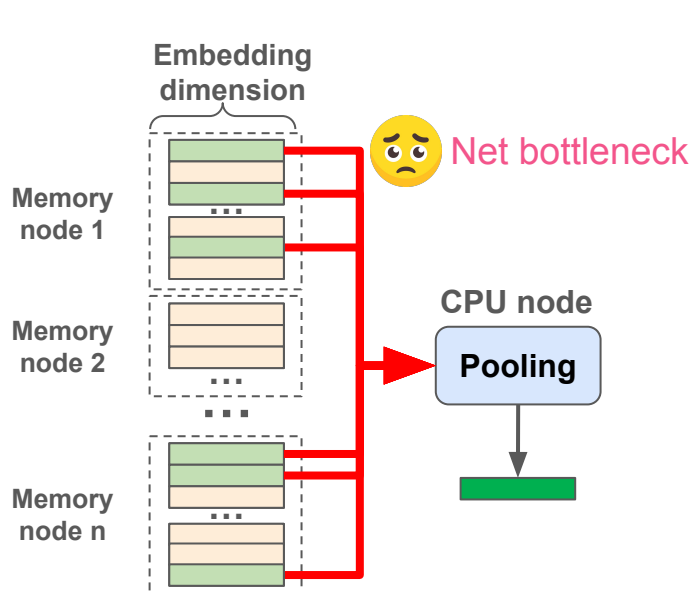
1a- CPU embedding cache using embedding **temporal locality**

- **Temporal locality:**
 - Some embeddings (e.g., 10%~15%) are frequently visited.
 - Some embeddings requested often appear together in the same lookup request.
- CPU node has limited memory resource (O(10) GB)
 - **Cache hot embeddings** by profiling the embedding hotness distributions of EMR models at runtime → Workload-aware caching.

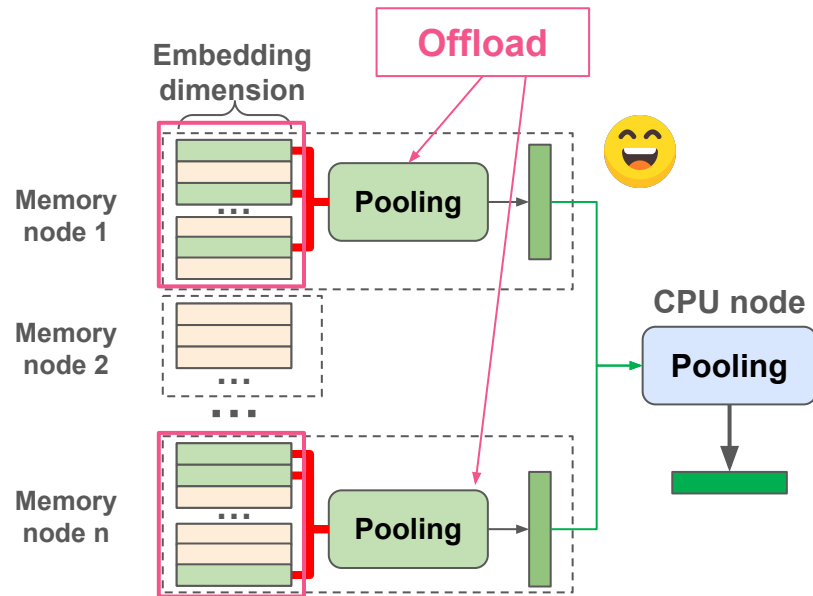
Design 1: Locality-aware embedding lookup

1b- Hierarchical embedding pooling leveraging spatial locality:

- **Spatial locality**: multiple embedding tables often co-locate in the same embedding server



(1) Strawman: embedding pooling



(2) Hierarchical embedding pooling

Preliminary results: Potential gain of locality-awareness

- FlexEMR effectively mitigates network bottlenecks
 - CPU embedding cache reduces **55%** embedding movement

Metric	Vanilla	+Cache
Emb Lookup Latency (μs)	5.7	5.42
Net BW Usage	100%	96%
# Embs over Net (M)	52	23.4

Preliminary results: Potential gain of locality-awareness

- FlexEMR effectively mitigates network bottlenecks
 - CPU embedding cache reduces **55%** embedding movement
 - Hierarchical embedding pooling further reduces latency by **24.1%**

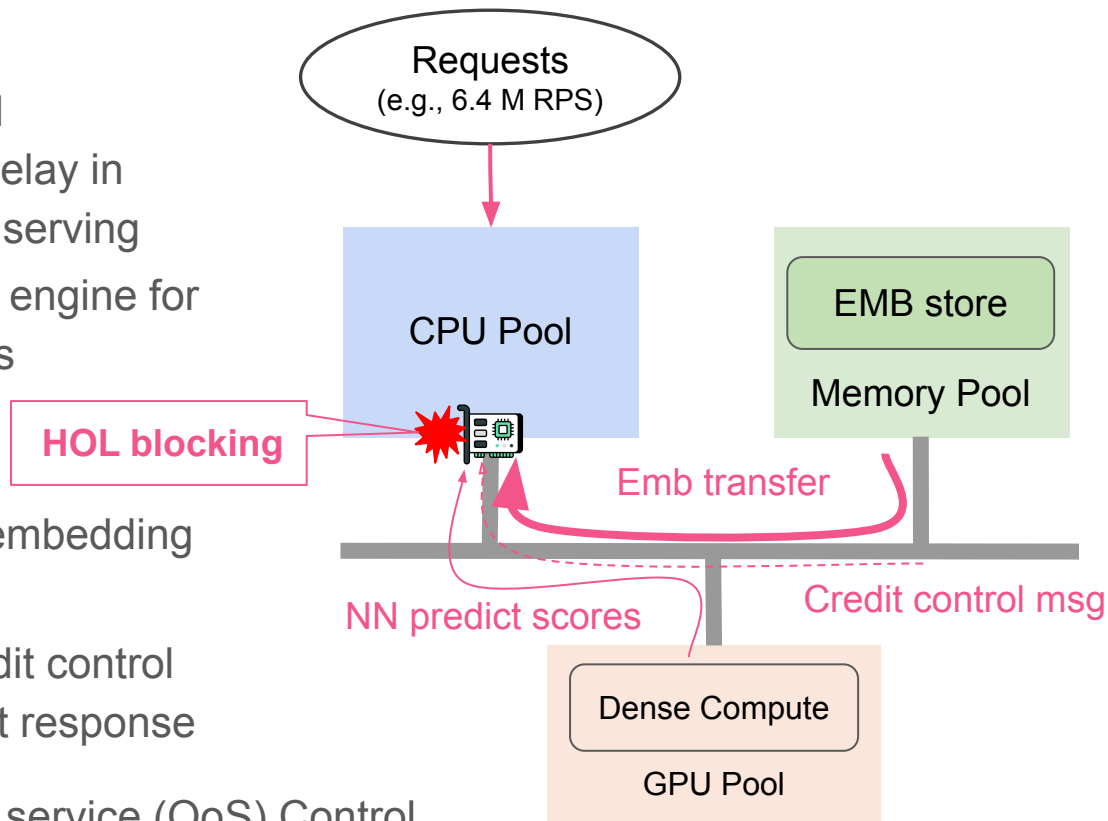
Metric	Vanilla	+Cache	FlexEMR
Emb Lookup Latency (μs)	5.7	5.42	4.11
Net BW Usage	100%	96%	80%
# Embs over Net (M)	52	23.4	19.4

Challenge 2: RDMA engine inefficiency

Existing RDMA engines:

- Single-threaded IO model
 - Increased queuing delay in disaggregated EMR serving
- ➔ Require more efficient IO engine for concurrent serving requests
- RNIC contentions
 - Bandwidth-intensive embedding transmission
 - Latency-sensitive credit control messages, NN predict response

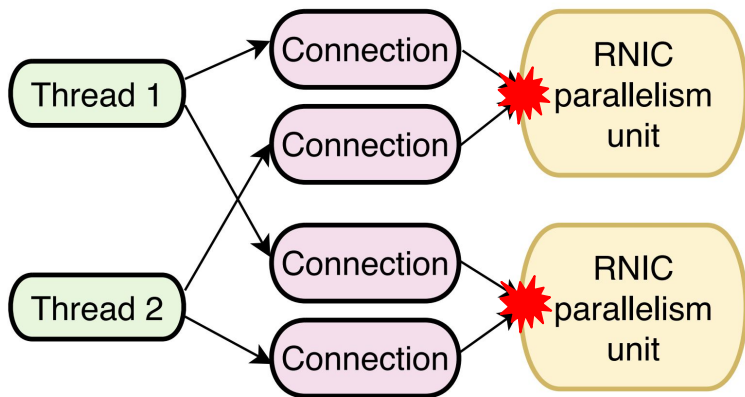
➔ Require RDMA quality of service (QoS) Control



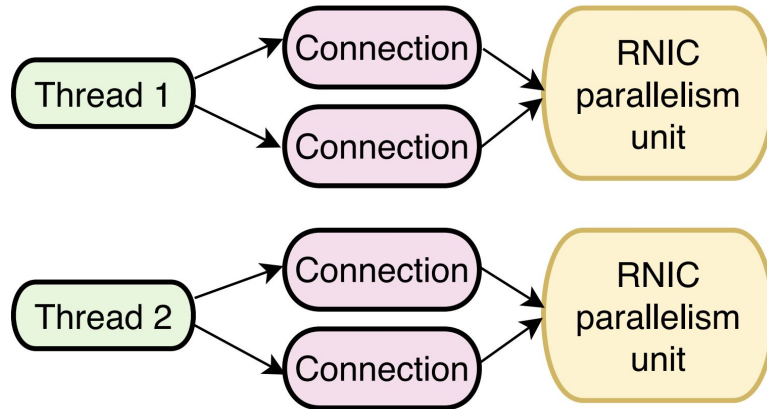
Design 2: RDMA-optimized communication

Key insight: Leverage multi-threaded RDMA with QoS control for concurrent requests.

- **2a- Multi-threaded RDMA engine**
 - Contention on RNIC resource.
 - **Context awareness** using RDMA resource domain technique



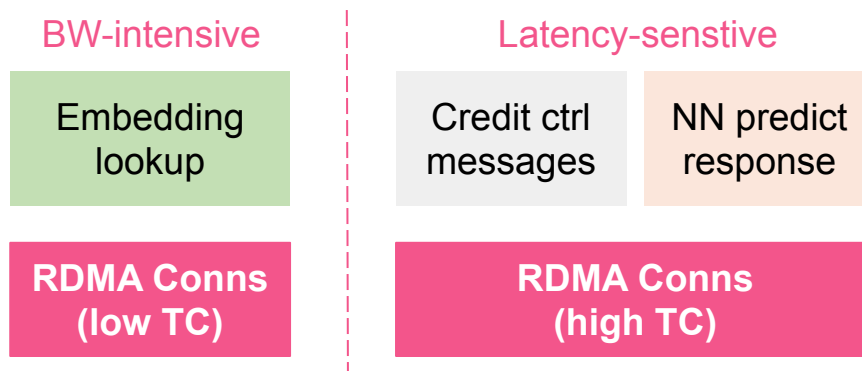
Default multi-threading



Context-aware multi-threading

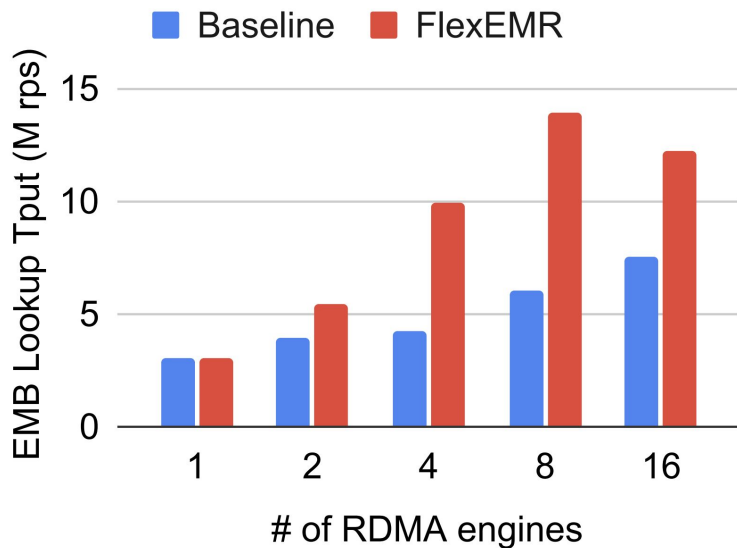
Design 2: RDMA-optimized communication

- **2b-** Hardware-based RDMA QoS flow channel to eliminate HOL blocking
 - Idea: Assign connections for credit control messages and NN predict response with higher traffic classes (TC).
 - Enabler: RDMA connection-level QoS support



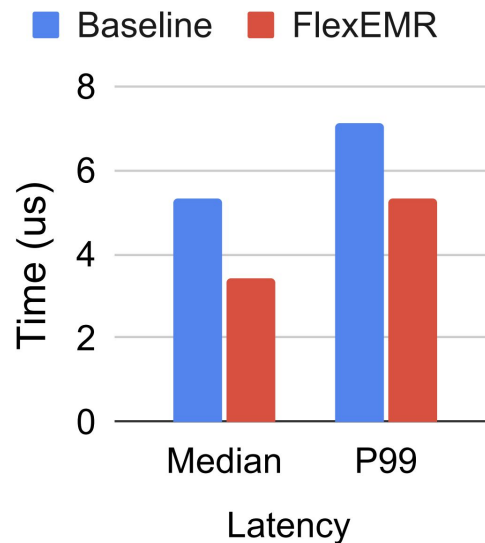
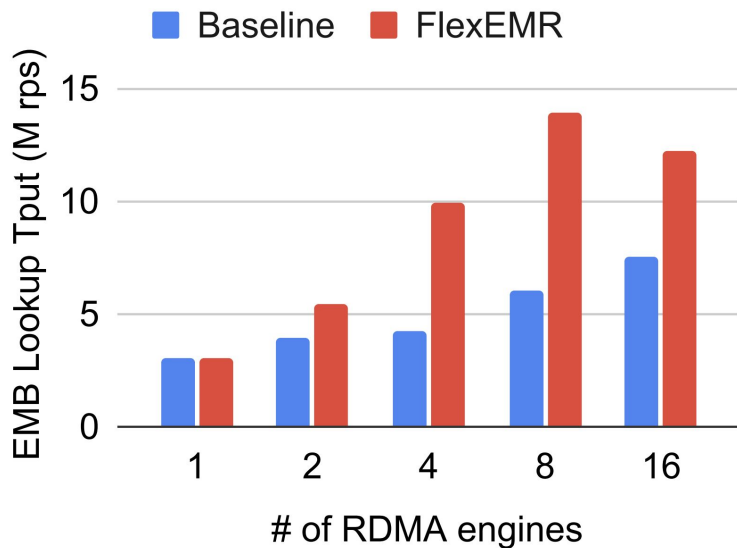
Preliminary results: Multi-threaded RDMA with QoS control

- FlexEMR outperforms naïve RDMA-based embedding lookup in efficiency
 - Efficient multi-thread RDMA improves throughput by up to **2.3 times**



Preliminary results: Multi-threaded RDMA with QoS control

- FlexEMR outperforms naïve RDMA-based embedding lookup in efficiency
 - Efficient multi-thread RDMA improves throughput by up to **2.3 times**
 - RDMA QoS flow channel reduces credit msg transmission latency by **35%**



Summary & Next Steps

- **Motivation:** EMR models evolve with variation in resource requirements, but serving them with fixed resource ratios leads to inefficiency and high cost.
- **FlexEMR:** Fully disaggregated recommendation model serving architecture.
- **Key insight:** Leverage model-network codesign principle to unlock the great potential of fully disaggregation.

Near-term goals

- Tracking embedding locality with low overhead.
- Enforcing resource scaling policies with high accuracy and efficiency.
- Extending FlexEMR benefits to other AI models that rely on embeddings.

ybhuang.cs@gmail.com