ETH zürich

Efficient Recommendation Inference on Heterogeneous CPU, GPU, FPGA Clusters

Wenqi Jiang

Systems Group, ETH Zurich

2023/06/23

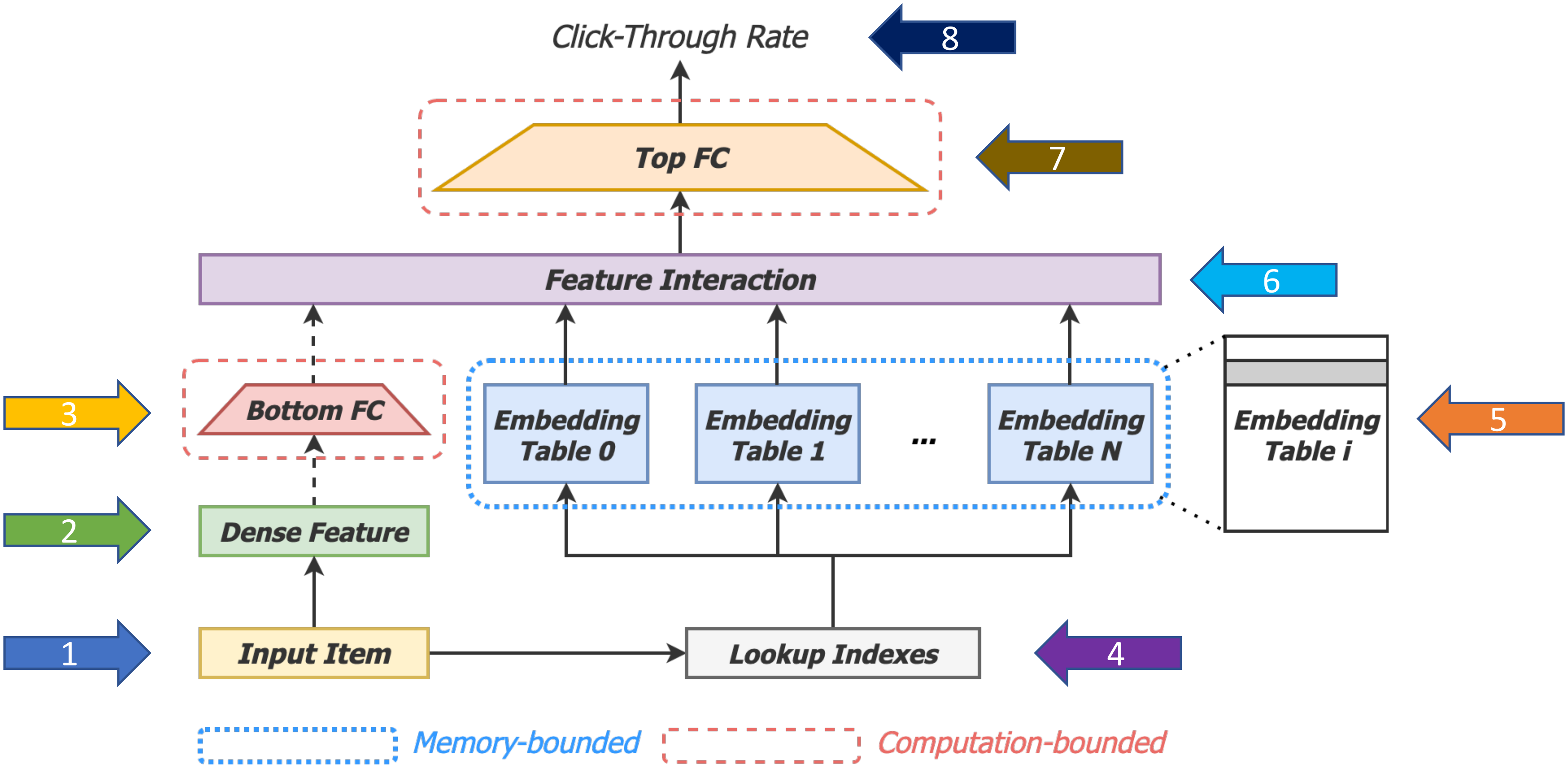
Alibaba Group

Systems@ETH zürich

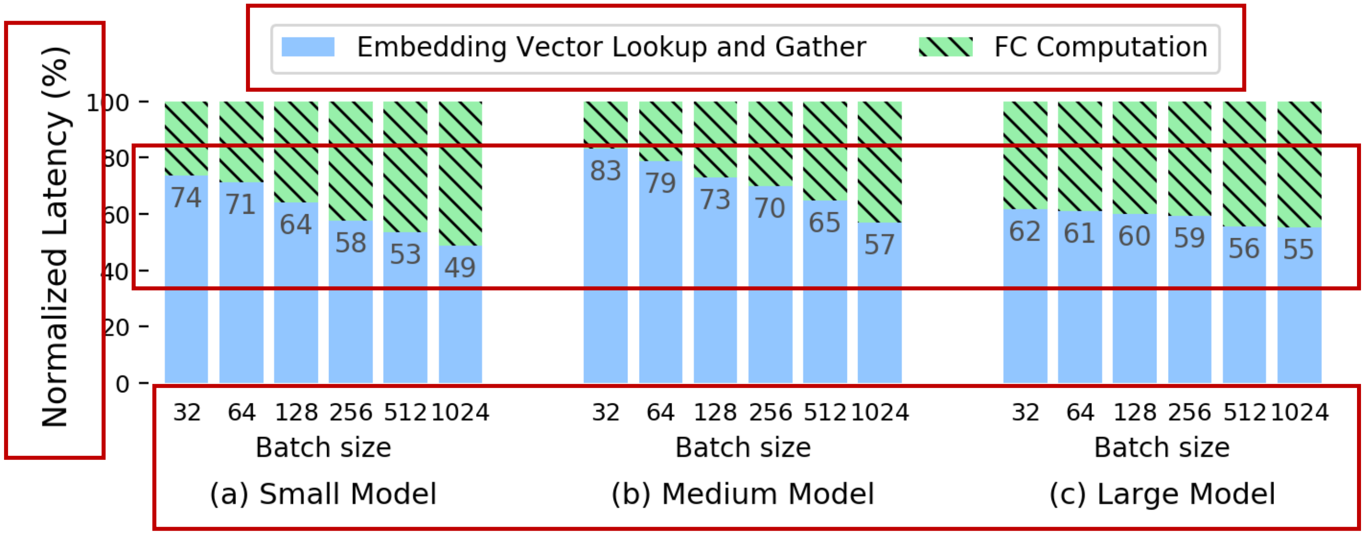
Personalized recommendation are everywhere

Up to 79% workload in data centers are recommendation inference!

Deep recommendation models involve intensive embedding table lookups



Workload profiling on Alibaba's real models



Embedding lookup comprises more than half of the inference

Why embedding table lookups are slow?

Many random DRAM accesses

- many embedding tables (tens to hundreds)
- each embedding vector is very short

Why embedding table lookups are slow?

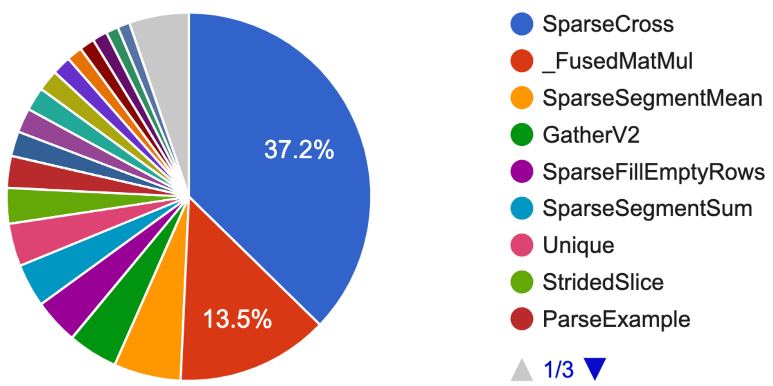
Existing ML frameworks are not optimized for embedding lookups

Function call overheads to preprocess inputs, retrieve the embedding vectors, and concatenate them together

TensoFlow Serving invokes 37 types of operators many times

ON HOST: TOTAL SELF-TIME (GROUPED BY TYPE)

(in microseconds) of a TensorFlow operation



An ideal recommendation inference system requires:

Fast embedding table lookups

Fast DNN computation

Support different model architectures

- different DNN layer parameters
- various table numbers (tens to hundreds)
- diverse model sizes (less than 1 GB to more than 1 TB)

GPUs are great for DNN computation, but not recommendation…

Small memory capacity

- cannot handle big models

Limited embedding table lookup performance

- many bank conflicts during random table lookups

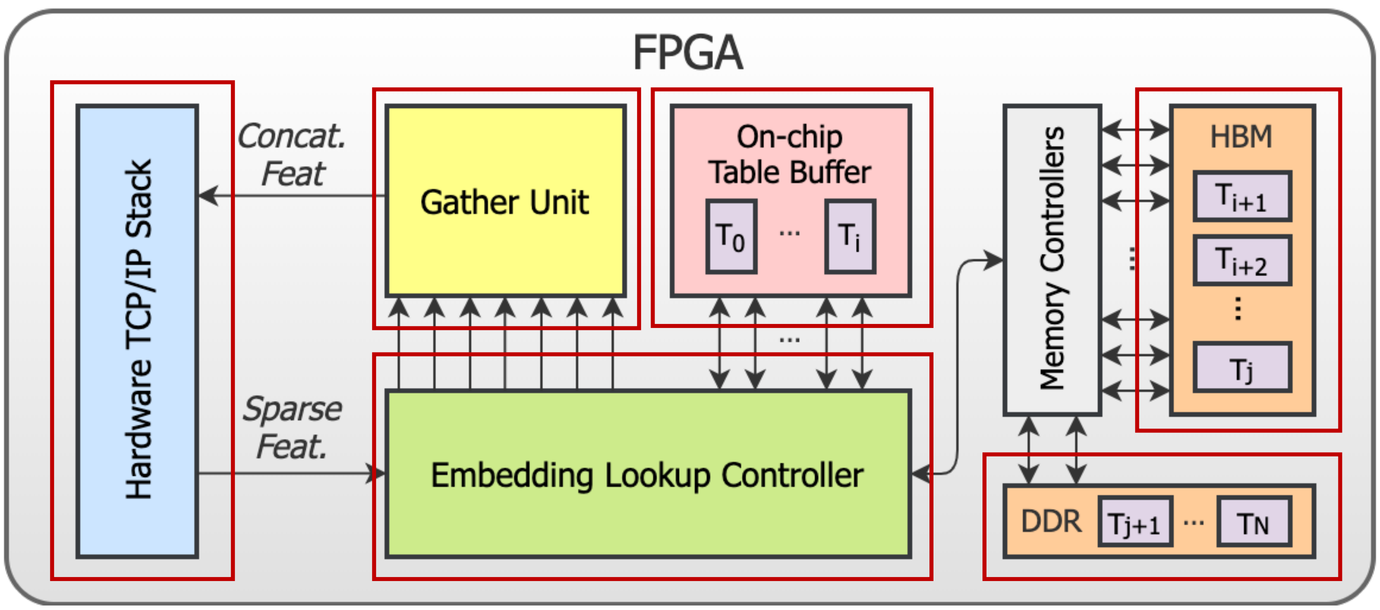
Latency concern

- require batching to maximize throughput
- SLA vs throughput

[1] Udit Gupta et al. “DeepRecSys: A System for Optimizing End-To-End At-scale Neural Recommendation Inference.” ISCA 2020
[2] Samuel Hsia et al. “Cross-Stack Workload Characterization of Deep Recommendation Systems.” IISWC 2020.
[3] Ranggi Hwang et al. “Centaur: A Chiplet-based, Hybrid Sparse-Dense Accelerator for Personalized Recommendations.” ISCA 2020.

How to design a great embedding lookup engine?

Without considering huge tables, an FPGA equipped with High-bandwidth Memory (HBM) is ideal



What about those models with huge tables?

An embedding table encoding user IDs can be huge

- one billion entries x 64-dimensional float vectors = 256 GB

Don’t need to store them in the expensive FPGA memory

- use DRAM on a regular CPU server
- even SSDs

Insight: take advantage of the strengths of multiple types of hardware

GPU for pure DNN computation

FPGA for accessing small and medium embedding tables

DRAM/SSD on CPU servers for huge embedding tables

Installing a certain number of GPUs and FPGAs on a same server is not the best idea

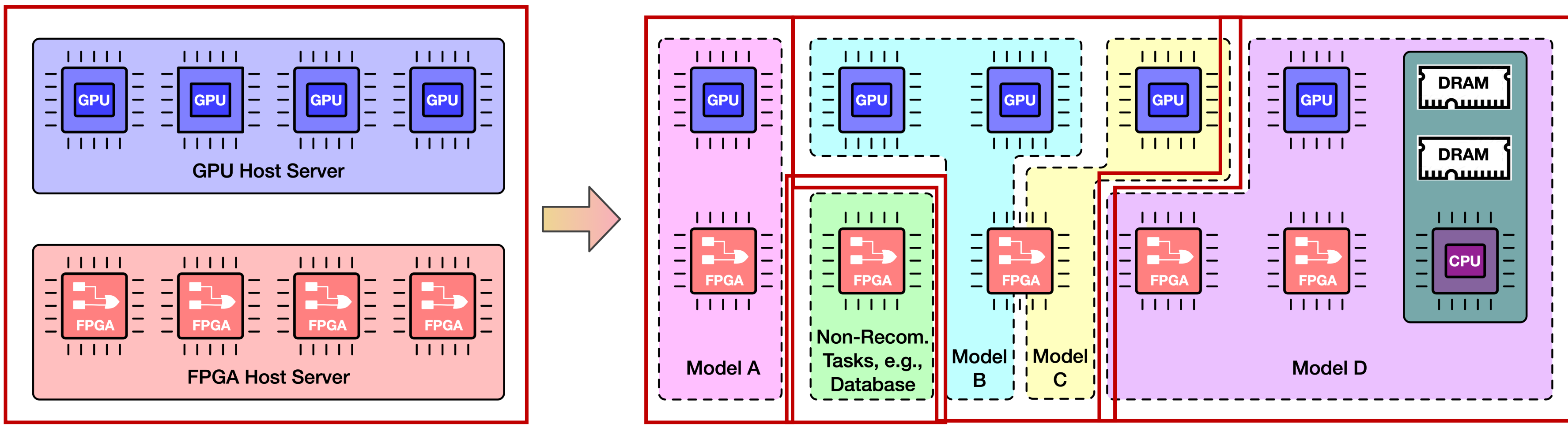
Couple 1 FPGA with 1 GPU is not always the best solution

Need special server for recommendation only

FleetRec: bridging CPUs, GPUs and FPGAs by network in the cloud

Using existing server

Flexible combination



Interconnect through network

Experiment Setup

Models

3 real-world models from Alibaba ranges from 1 GB ~ 100+ GB

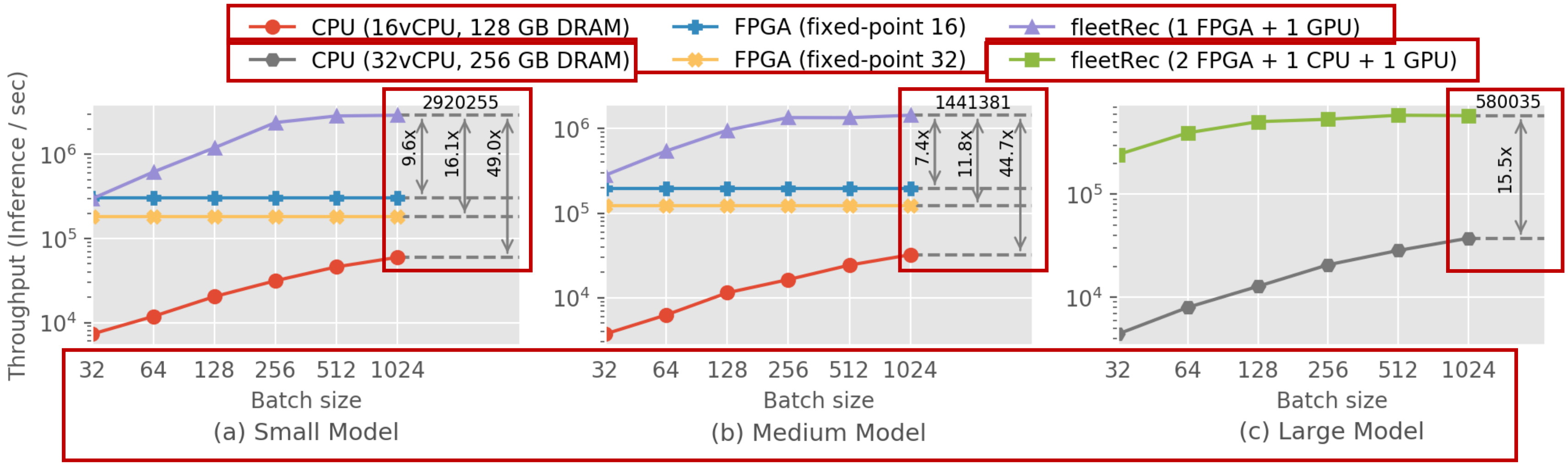
Hardware

FPGA: Xilinx Alveo U280: 8 GB HBM + 32 GB DDR4

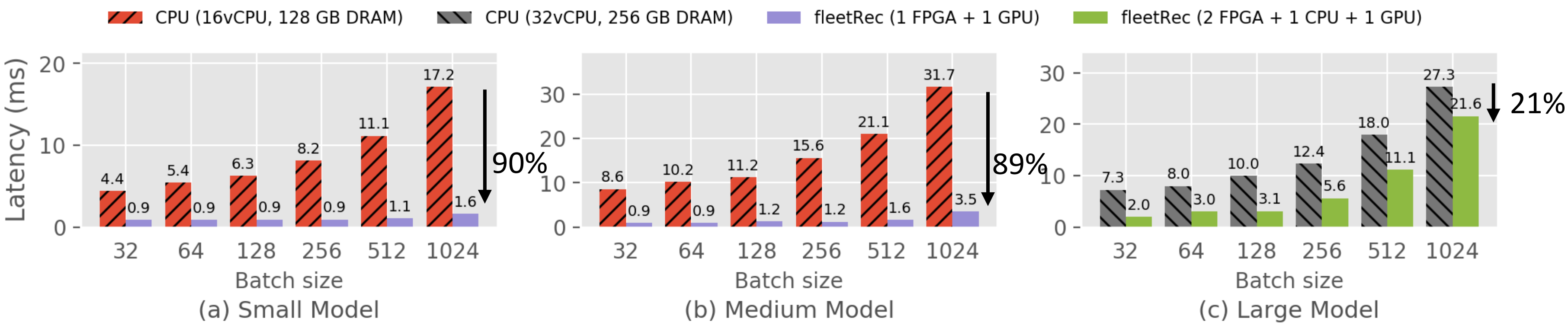
GPU: NVIDIA Titan RTX

CPU baseline: Intel Xeon E5-2686 v4 CPU @2.30GHz (16~32 vCPU); 128~256 GB DDR4 (8 channels); TensorFlow Serving

FleetRec achieves significant throughput speedup over CPU / FPGA baseline



FleetRec is also better in terms of latency compared with CPU



For real-time recommendation with latency constraints, FleetRec is more advantageous

	Small Model			Medium Model			Large Model		
SLA (ms)	5	10	20	5	10	20	5	10	20
Throughput (inferences / sec)									
CPU	7.30E+3	3.14E+4	5.96E+4	N/A	3.72E+3	1.64E+4	N/A	1.28E+4	2.85E+4
FPGA	3.05E+5	3.05E+5	3.05E+5	1.95E+5	1.95E+5	1.95E+5	N/A	N/A	N/A
FleetRec	2.92E+6	2.92E+6	2.92E+6	1.44E+6	1.44E+6	1.44E+6	5.07E+5	5.35E+5	5.80E+5
Speedup of FleetRec over									
FPGA	9.57×	9.57×	9.57×	7.39×	7.39×	7.39×	+∞×	+∞×	+∞×
CPU	400.07×	92.97×	48.96×	+∞×	387.24×	87.92×	+∞×	41.76×	20.34×

Concluding remarks

Deep recommendation model contains:

- many embedding lookups
- DNN computation

FleetRec: a high-performance recommendation inference system on heterogeneous hardware

- takes advantages of the strengths of each type of hardware
- interconnect different hardware by network for flexible combination