Hercules: Heterogeneity-Aware Inference Serving for At-Scale Personalized Recommendation

Liu Ke*, Udit Gupta**, Mark Hempstead◇, Carole-Jean Wu*,
Hsien-Hsin S. Lee*, Xuan Zhang▲

*Meta, ▲Washington University in St. Louis, ◇Harvard University, ◇Tufts University
Overview

• Motivation

• Background

• Proposed Design: Hercules

• Performance Evaluation

• Conclusion
Nowadays, personalized recommendation is a fundamental building block of many internet services.
DL-based Recommendation Models

- Deep learning (DL-)based recommendation model consists of
  - dense features processed by fully-connected (FC) layers
  - sparse features processed by indexing embedding tables, implemented as SparseLengthsSum (SLS) operator in Caffe2

Contiguous (Dense) Features
  - User Age
  - User Friends Count
  - Item Click Ratio

Categorical (Sparse) Features
  - User Click History (Multi-hot)
  - Item Category (One-hot)

Embedding Tables
  - Gather
  - Reduce

Bottom FC

Dense & Sparse Features Interaction

Predict FC

Click Through Rate
  - 84%
  - 71%
  - 95%
  - 65%
  - 52%
Challenges of Inference Serving at-Scale

- **Model Diversity**
  - Construct differently for a wide variety of services
  - Evolve rapidly for higher prediction accuracy
Challenges of Inference Serving at-Scale

- **Time-varying Load Patterns**
  - The query sizes arriving individual servers exhibit a heavy-tail distribution
    - Constraint: strict tail-latency target set by Service Level Agreement (SLA)
  - The diurnal loads arriving the cluster exhibit highly-fluctuating & synchronous patterns
    - Constraint: global throughput target

![Inference Query Size Distribution](chart)

- **Individual server level**
  - Heavy-tail of query sizes
  - Strict SLA tail-latency target

- **Cluster level**
  - Variations across different data centers (DC-1 to DC-4)
Challenges of Inference Serving at-Scale

- **Cloud-scale System Heterogeneity**
  - System upgrades occur periodically
  - Domain-specific accelerators are increasingly deployed in datacenters

![Diagram showing different server types and accelerators](image)

- Intel Xeon CPUs
  - Broadwell
  - Skylake
  - Cooper lake

- Nvidia GPUs
  - P100
  - V100
  - A100

- Google TPU

- Alibaba Hanguang

- Processing-in/near Memory

- Server Type I
  - CPU

- Server Type II
  - CPU
  - NMP PU

- Server Type III
  - CPU
  - PCIe
  - Accelerator

……
Overview

• Motivation

• Background

• Proposed Design: Hercules

• Performance Evaluation

• Conclusion
Background: System Stack

- System stack consists of task scheduler, DL framework, underlying hardware architecture
- Task scheduler exploits data-, model-, and operator-parallelism
Background: Cluster Management

- Workload classification: rank the workloads’ performance on the different server architectures
- Scheduling policy: heterogeneity-oblivious scheduler and greedy scheduler in [1][2]

---

[1] Christina Delimitrou, Christos Kozyrakis, “Paragon: QoS-aware scheduling for heterogeneous datacenters,” in ASPLOS, 2013
[2] Christina Delimitrou, Christos Kozyrakis, “Quasar: Resource-Efficient and QoS-Aware Cluster Management,” in ASPLOS, 2014
Overview

• Motivation

• Background

• Proposed Design: Hercules

• Performance Evaluation

• Conclusion
SLA-aware Task Scheduling

- Latency-critical recommendation workloads must satisfy the strict SLA latency target
- Hercules proposes gradient-based search to identify the optimal task scheduling configuration
Heterogeneity-aware Provisioning

- Why consider heterogeneity at cluster level?
  - Up to 30x performance and 6x energy efficiency variation
- Offline profiling
  - Measure and record $QPS_{H,1:M}$ and $Power_{H,1:M}$ for accurate workload classification

Proposed Design: Hercules
Proposed Design: Hercules

Heterogeneity-aware Provisioning

- Hercules formulates the provisioning as a constrained optimization problem
- Online serving
  - Calculate $N_{1:H,1:M}(t)$ with standard linear optimization solver, e.g. simplex, interior-point

The number of server type $T_h$ assigned to workload $G_m$

\[
\text{Minimize } \sum_{m=1}^{M} \left( \sum_{h=1}^{H} (N_{h,m}(t) \times \text{Power}_{h,m}) \right) \quad \text{(1), subject to}
\]

\[
\forall m \in [1..M], \sum_{h=1}^{H} (N_{h,m}(t) \times QPS_{h,m}) \geq \text{load}_m(t)(1+R\%) \quad \text{(2)}
\]

\[
\forall h \in [1..H], \sum_{m=1}^{M} N_{h,m}(t) \leq N_h \quad \text{(3)}
\]
Heterogeneity-aware Provisioning

- Hercules formulates the provisioning as a constrained optimization problem
- Online serving
  - Calculate $N_{1:H,1:M}(t)$ with standard linear optimization solver, e.g. simplex, interior-point

Objective: the total amount of provisioned power

\[
\text{Minimize} \sum_{m=1}^{M} \left( \sum_{h=1}^{H} (N_{h,m}(t) \times \text{Power}_{h,m}) \right) \quad (1) \quad \text{subject to}
\]

\[
\forall m \in [1..M], \sum_{h=1}^{H} (N_{h,m}(t) \times QPS_{h,m}) \geq \text{load}_m(t)(1+R\%)(2),
\]

\[
\forall h \in [1..H], \sum_{m=1}^{M} N_{h,m}(t) \leq N_h \quad (3)
\]
Heterogeneity-aware Provisioning

- Hercules formulates the provisioning as a constrained optimization problem
- Online serving
  - Calculate $N_{1:H,1:M}(t)$ with standard linear optimization solver, e.g. simplex, interior-point

Constraint: the activated servers can handle the incoming loads

Time interval adjusting the provisioning
Proposed Design: Hercules

Heterogeneity-aware Provisioning

- Hercules formulates the provisioning as a constrained optimization problem
- Online serving
  - Calculate $N_{1:H,1:M}(t)$ with standard linear optimization solver, e.g. simplex, interior-point

$$\text{Minimize } \sum_{m=1}^{M} \left( \sum_{h=1}^{H} (N_{h,m}(t) \times \text{Power}_{h,m}) \right) \quad (1), \text{ subject to}$$

$$\forall m \in [1..M], \sum_{h=1}^{H} (N_{h,m}(t) \times QPS_{h,m}) \geq \text{load}_m(t)(1+R\%) \quad (2),$$

$$\forall h \in [1..H], \sum_{m=1}^{M} N_{h,m}(t) \leq N_h \quad (3)$$

Constraint: the activated servers are not exceeding the total available servers.
Overview

- Motivation
- Background
- Proposed Design: Hercules
- Performance Evaluation
- Conclusion
Performance Evaluation

- Synthetic model evolution
  - Linearly varying the composition of the workloads
  - On CPU-only cluster, the snapshots on Day-D1 and Day-D2
    - Cluster Capacity: 2.3x at peak
    - Provisioned Power: 1.8x at peak

Cluster Configuration

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| $T_h$ | $N_h$ | CPU | Memory | $T_h$ | $N_h$ | CPU | Memory | GPU |
| $T_1$ | 100  | CPU-T1 | DDR4  | $T_6$ | 10   | CPU-T1 | DDR4  | P100 |
| $T_2$ | 100  | CPU-T2 | DDR4  | $T_7$ | 5    | CPU-T2 | DDR4  | V100 |
| $T_3$ | 15   | CPU-T2 | NMPx2 | $T_8$ | 6    | CPU-T2 | NMPx2 | V100 |
| $T_4$ | 10   | CPU-T2 | NMPx4 | $T_9$ | 4    | CPU-T2 | NMPx4 | V100 |
| $T_5$ | 5    | CPU-T2 | NMPx8 | $T_{10}$ | 2  | CPU-T2 | NMPx8 | V100 |

- Deploy accelerated servers ($T_3$-$T_{10}$) in the cluster
Performance Evaluation

- Comparison with prior cluster schedulers
  - SOTA greedy scheduler vs. heterogeneity-oblivious (NH) scheduler
    - 76% capacity saving and 51% provisioned power saving at peak
  - Hercules scheduler vs. greedy scheduler
    - 48% capacity saving and 24% provisioned power saving at peak
Overview

• Motivation

• Background

• Proposed Design: Hercules

• Performance Evaluation

• Conclusion
Conclusion

• Challenges of Inference Serving at-Scale
  ▪ Model Diversity
  ▪ Time-varying Load Patterns
  ▪ Cloud-scale System Heterogeneity

• Proposed Hercules Design
  ▪ SLA-aware Task Scheduling
  ▪ Heterogeneity-aware Provisioning

• Hercules achieves up to 48% capacity saving and 24% provisioned power saving over a SoTA greedy scheduler

Contact: ke.l@wustl.edu
This presentation and recording belong to the authors. No distribution is allowed without the authors’ permission