AI Scale at the Modern Era

Carole-Jean Wu & Niket Agarwal
Facebook
What Drove the Deep Learning Era?

A Virtuous Cycle

BETTER ALGORITHMS

MORE COMPUTE

BIGGER (AND BETTER) DATA
Machine Learning at Facebook

- Machine learning is used extensively
  - Ranking posts
  - Content understanding
  - Object detection, segmentation, and tracking
  - Speech recognition/translation
- From data centers to the edge

Machine Learning Execution Flow

1. Data Preprocessing
2. Training
3. Inference

DATA → FEATURES → TRAINING → EVALUATION → DEPLOYMENT
Data Scale at Facebook (and elsewhere)

- XXX PB replicated daily
- XX PB ingested daily
- X TB/s stream processing throughput
- XXX PB daily shuffle
- XM Machines
- X EB Warehouse Size
- XX K pipelines
- X K pipeline authors
Diversity in DL Use Cases

Must not over-design for GEMM nor convolutions

Flexibility requires generality
Training Data and Feature Growth for Recommender Systems

**Data Storage Growth**

Training data for recommendation models has grown by 1.75x in 2 years

**Model Memory Growth**

Size of Facebook’s production recommendations models has grown by an order of magnitude in 3 years

---

ML Trends
- Data explosion
- Freshness & latency
- Standardization
- Privacy and security
- Complex data models
- Richer query methods

System Trends
- Disaggregation
- Horizontal scaling
- Elastic compute
- Power efficiency
- Global optimization
- Engineering efficiency
Machine Learning Execution Flow

1. Data Preprocessing
2. Training
   - MLPPerf Training
3. Inference
   - MLPPerf Inference

Data Ingestion Pipeline
Data Ingestion Pipeline
A Typical Data Ingestion Pipeline for MLPerf

Cloud Storage → Dataset downloaded to local storage → Local Storage → Host CPU → Preprocessed tensors loaded onto GPUs → Training GPUs

- Raw batches read from local storage
ML Training Storage growth @FB

~1.75x growth in training data storage size over past 2 years

~13x growth in training data ingestion throughput projected over 3 years
ML Training datasets cannot be stored locally on Trainers

<table>
<thead>
<tr>
<th>Model</th>
<th>Table Size (PB)</th>
<th>Partition Size (PB)</th>
<th>Used Partition Size (PB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM1</td>
<td>13.45</td>
<td>0.15</td>
<td>11.95</td>
</tr>
<tr>
<td>RM2</td>
<td>29.18</td>
<td>0.32</td>
<td>25.94</td>
</tr>
<tr>
<td>RM3</td>
<td>2.93</td>
<td>0.07</td>
<td>1.95</td>
</tr>
</tbody>
</table>
ML Training Preprocessing @FB

<table>
<thead>
<tr>
<th>Modelz</th>
<th>kQPS</th>
<th>Storage RX (GB/s)</th>
<th>Transform RX (GB/s)</th>
<th>Transform TX (GB/s)</th>
<th># CPU Sockets required</th>
</tr>
</thead>
<tbody>
<tr>
<td>RM1</td>
<td>11.623</td>
<td>0.8</td>
<td>1.37</td>
<td>0.68</td>
<td>24.16</td>
</tr>
<tr>
<td>RM2</td>
<td>7.995</td>
<td>1.2</td>
<td>0.96</td>
<td>0.50</td>
<td>9.44</td>
</tr>
<tr>
<td>RM3</td>
<td>36.921</td>
<td>0.8</td>
<td>1.01</td>
<td>0.22</td>
<td>55.22</td>
</tr>
</tbody>
</table>

ML training preprocessing compute requirements exceed trainer host capabilities
Local data storage and preprocessing doesn’t work for us!

Cloud Storage → Dataset downloaded to local storage → Local Storage → Raw batches read from local storage → Host CPU → Preprocessed tensors loaded onto GPUs → Training GPUs

NVIDIA DGX
Disaggregated Training Data Ingestion @FB
**Disaggregated Training Data Ingestion @FB**

<table>
<thead>
<tr>
<th>Event Logs</th>
<th>Offline ETL Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML Engineer</td>
<td>Job Config.</td>
</tr>
</tbody>
</table>

**Data Warehouse Table Partition**

<table>
<thead>
<tr>
<th>Float</th>
<th>ID List</th>
<th>ID Score List</th>
</tr>
</thead>
<tbody>
<tr>
<td>map&lt;int, float&gt;</td>
<td>map&lt;int, list&lt;int&gt;&gt;</td>
<td>map&lt;int, map&lt;int, float&gt;&gt;</td>
</tr>
</tbody>
</table>

**Distributed Pte-Processing**

- **DPP Master Node**
  - Checkpointing
  - Monitoring
- **DPP Worker Nodes**
  - Extract
  - Raw Rows
  - Transform
  - Tensors

**Training Frontend Framework**

**Training Cluster**

- **Training Cluster Master**
  - Checkpointing
  - Monitoring
- **Training Cluster Nodes**
  - DPP Client
  - PyTorch
  - CUDA / GPU

**Blob Store**

**Storage tier**

**Reader tier**

**Trainers**
Disaggregated Training Data Ingestion @FB

Storage tier

Reader tier

Trainees
Disaggregated Training Data Ingestion @FB

Training Frontend Framework

Distributed Pre-Processing

DPP Master Node
- Checkpointing
- Monitoring

DPP Worker Nodes
- Extract
- Raw Rows
- Transform
- Tensors
- Server
- Tensors

Storage tier

Reader tier

Trainers

Event Logs
- ML Engineer
- Job Config.

Offline ETL Job

Data Warehouse Table Partition
- Float
- ID List
- ID Score List
- ...
**Disaggregated Training Data Ingestion @FB**

**Storage tier**

**Reader tier**

**Trainers**
Disaggregation is not enough: Training Data Ingestion Challenges

Data ingestion (Storage + Preprocessing) represents a significant, and growing, component of training capacity.
End-to-end Co-design for Data Ingestion Efficiency

Feature Flattening
  • Push data filtering to storage nodes

In-Memory Flatmap
  • Optimized data formats

Merged Reads
  • Increased disk throughput

Feature Reordering
  • Mitigate unnecessary reads
## Regular Map Reads

<table>
<thead>
<tr>
<th>Row idx</th>
<th>Features (map&lt;str: int&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A: 1, B:1, C:3, D:1, E:3, F:3</td>
</tr>
<tr>
<td>2</td>
<td>A: 2, B:1, C:2, D:1, E:2, F:6</td>
</tr>
</tbody>
</table>

### Hive Table

- **Row idx 1**: 
  - A: 1
  - B: 1
  - C: 3
  - D: 1
  - E: 3
  - F: 3

- **Row idx 2**: 
  - A: 2
  - B: 1
  - C: 2
  - D: 1
  - E: 2
  - F: 6

### Read Features (A, D)

- **Read**: A: 1, B: 1, C: 3, D: 1, E: 3, F: 3
- **Entire rows are read**: A: 2, B: 1, C: 2, D: 1, E: 2, F: 6
# Feature Flattening

## Hive Table

<table>
<thead>
<tr>
<th>Row idx</th>
<th>Features (map&lt;str: int&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A: 1, B:1, C:3, D:1, E:3, F:3</td>
</tr>
<tr>
<td>2</td>
<td>A: 2, B:1, C:2, D:1, E:2, F:6</td>
</tr>
</tbody>
</table>

## Read Features (A, D)

A, D read via multiple seeks
Feature Flattening + Merged Reads

Hive Table

<table>
<thead>
<tr>
<th>Row idx</th>
<th>Features (map&lt;str: int&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A: 1, B:1, C:3, D:1, E:3, F:3</td>
</tr>
<tr>
<td>2</td>
<td>A: 2, B:1, C:2, D:1, E:2, F:6</td>
</tr>
</tbody>
</table>

Read Features (A, D)

One seek reads A and D, but over-reads B and C
Feature Flattening + Merged Reads + Feature Reordering

Hive Table

<table>
<thead>
<tr>
<th>Row idx</th>
<th>Features (map&lt;str: int&gt;)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A: 1, B:1, C:3, D:1, E:3, F:3</td>
</tr>
<tr>
<td>2</td>
<td>A: 2, B:1, C:2, D:1, E:2, F:6</td>
</tr>
</tbody>
</table>

Read Features (A, D)
One seek reads A and D
Training Data Efficiency Impact through co-design

2X power and cost savings for Data Ingestion
Future Opportunities: Training Data Reuse and Flash Caching

A subset of bytes (20-40%) contribute to most of Storage IO

Opportunity for Flash to absorb the IO more efficiently
Machine Learning Execution Flow

1. Data Manipulation
2. Training
   - MLPerf Training
3. Inference
   - MLPerf Inference

Data Ingestion Pipeline
High System Diversity for ML at the Edge

The diversity of mobile hardware and software is not found in the controlled datacenter environment.
Significant In-the-Field Performance Variability
Conclusion

Ever-Increasing AI Growth

Diverse ML System Requirement

Compute, Memory, Networking
Thank You