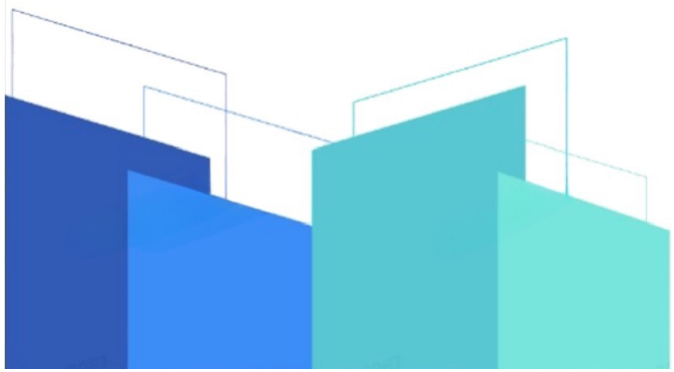
USENIX ATC '25



Primus: Unified Training System for Large-Scale Deep Learning Recommendation Models

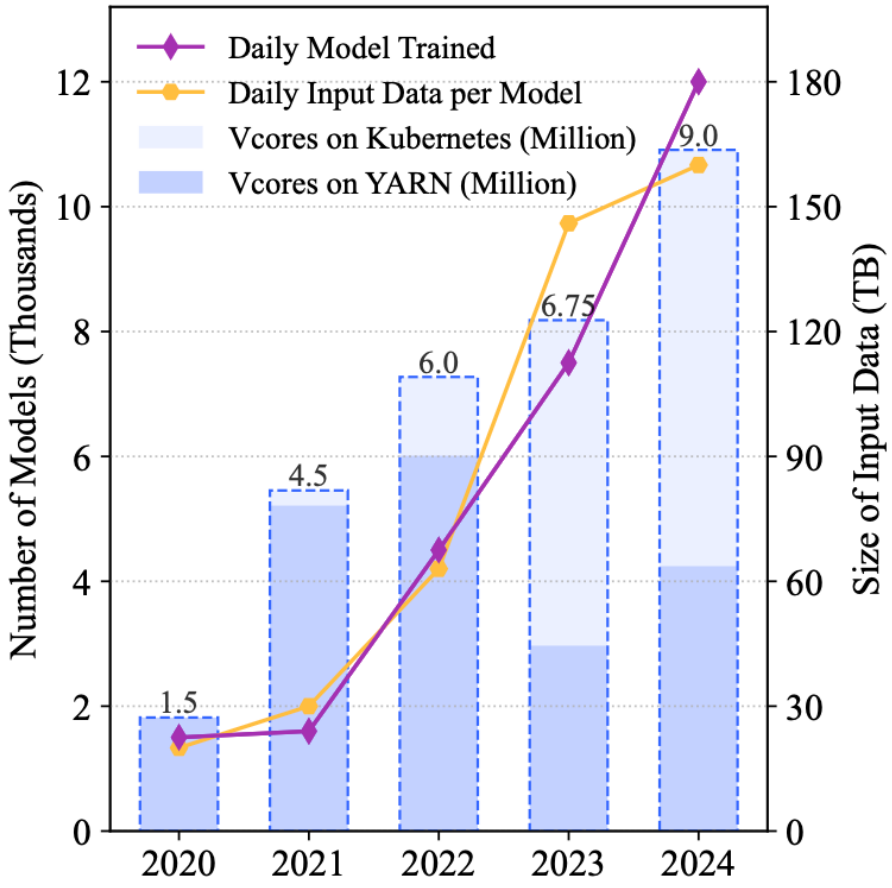
Jixi Shan†, Xiuqi Huang‡, Yang Guo†, Hongyue Mao†, Ho-Pang Hsu†, Hang Cheng†, Can Wang†, Jun Song†, Rui Shi†, Xiaofeng Gao§, Jingwei Xu†, Shiru Ren†, Jiaxiao Zheng†, Hua Huang†, Lele Yu†, Peng Xu†, Guihai Chen§

†ByteDance Inc. ‡State Key Lab of CAD&CG, Zhejiang University §Shanghai Jiao Tong University

Presenter: Jixi Shan

July 8, 2025

Deep Leaning Recommendation Model Workloads



Our DLRM Trends

- Have **grown tenfold** in five years
- **12,000** DLRM models trained daily
- **20TB → 160TB** average training data per model daily
- **1.5M → 9M** CPU virtual cores, deployed on both YARN and Kubernetes

Now

20M CPU vCores
200K GPUs
7EB Data

ByteDance 字节跳动

Challenges #1 Complex and diverse industrial training resources

- **Diverse Resource Env: YARN vs. Kubernetes**
  - ByteDance use lots of hybrid resources (Overall 20M vCores)
    - ***Problem****: DLRM Model Training Jobs Mixed with Online Services or Big Data Jobs*





kubernetes

- **Elastic Training across diverse resource environments**
  - HPA: Horizontal Pod Autoscaler *(Dynamic Change Parallism)*
    - ***Problem:*** *Slow reaction time for large scale cluster*
  - VPA: Vertical Pod Autoscaler *(Memory, CPU adjustment)*
    - ***Problem****: Elastic Scheduling should implement many times*

ByteDance字节跳动

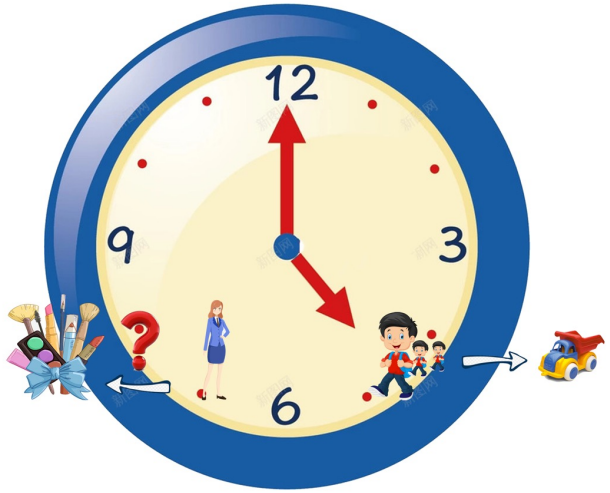
Challenges #2 Large-scale multi-source batch and stream training data

- **Preprocessing: training sample preprocessing tasks consume increasing resources**
  - *e.g., Feature engineering Spark jobs to merge multiple data sources*
- **Efficiency: Training dataset volume continues to grow rapidly**
  - *e.g., 20TB → 160TB*
- **Complexity: Diverse data storages and file formats (Hadoop-compatible)**
  - *e.g., HDFS(TF Records, Parquet), Feature Store (Iceberg, Hudi), Kafka*
- **Usability: One framework support for both batch and stream processing**

ByteDance字节跳动

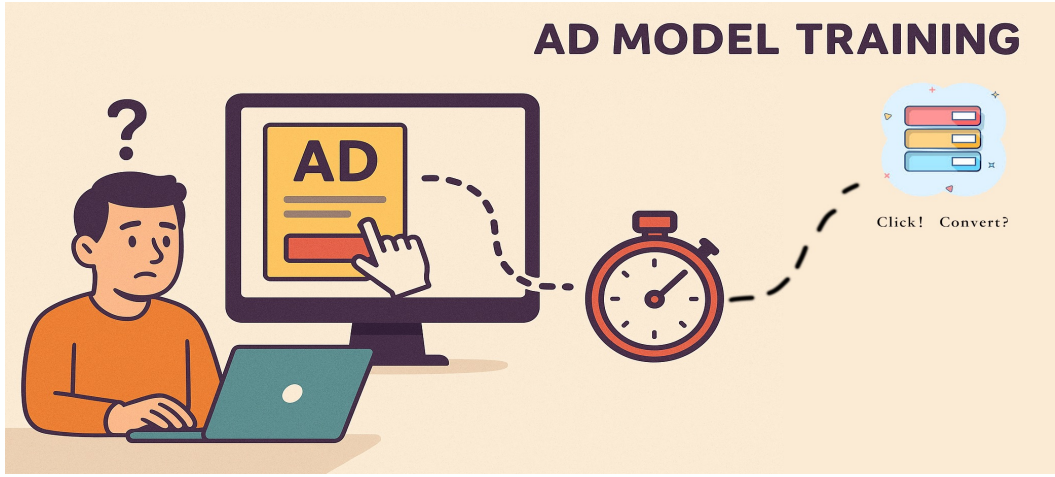
Challenges #3 Hard to improve online (real time) model effectiveness

- Catastrophic forgetting



- *How to improve the model performance for office staff after the model is updated with lots of school student instances?*

- Delayed conversion feedback



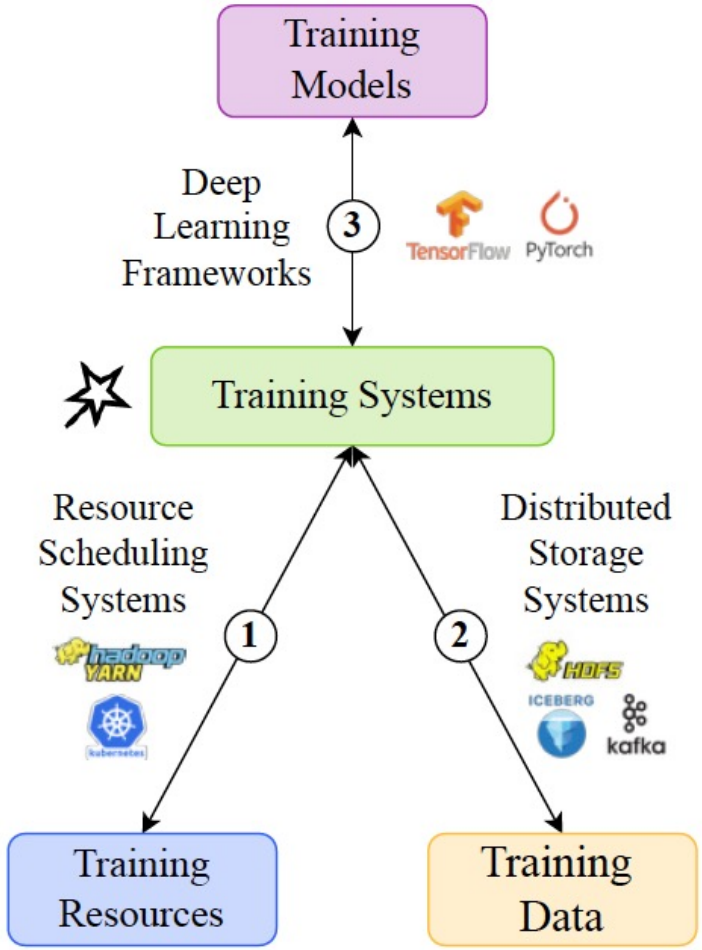
- *How to handle the delayed feedback*

ByteDance字节跳动

Overview: Unified Training Systems

Primus demonstrates its efficiency and effectiveness in handling large-scale, enterprise-grade DLRM training

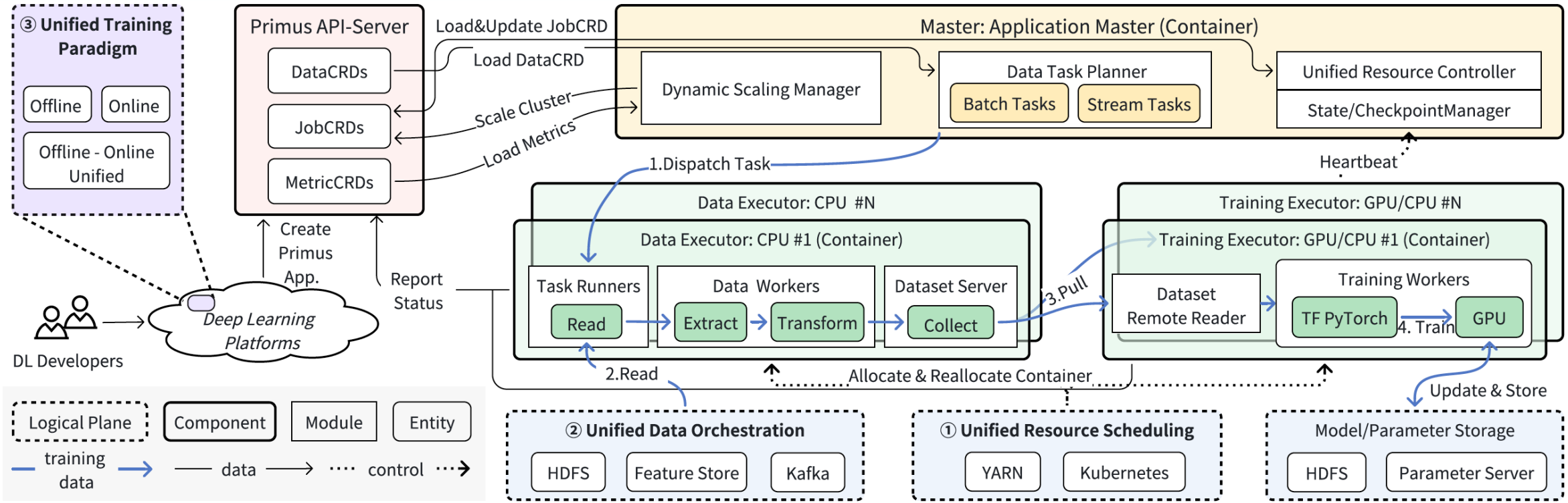
- Resource Scheduling: Horizontal scaling saves over 17.1% of resources while vertical scaling improves utilization from 50% to 80%.
- Data Orchestration: Generating around 4 million data tasks speeds up 23× in 1 thread and can be completed in 42s with 4 threads.
- Training Paradigm: Model effectiveness in advertising is significantly improved by 0.03%-0.07% in AUC and 0.4%-2.4% in revenue.



ByteDance字节跳动

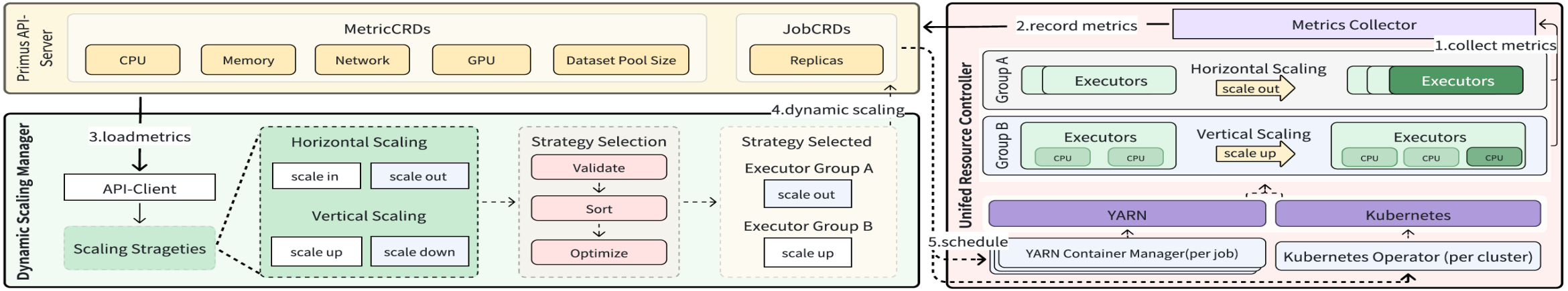
Overview: A typical DLRM training process with Primus

- Submit → Primus API-Server → Master → CPU Executors & GPU Executors →Update & Store



ByteDance 字节跳动

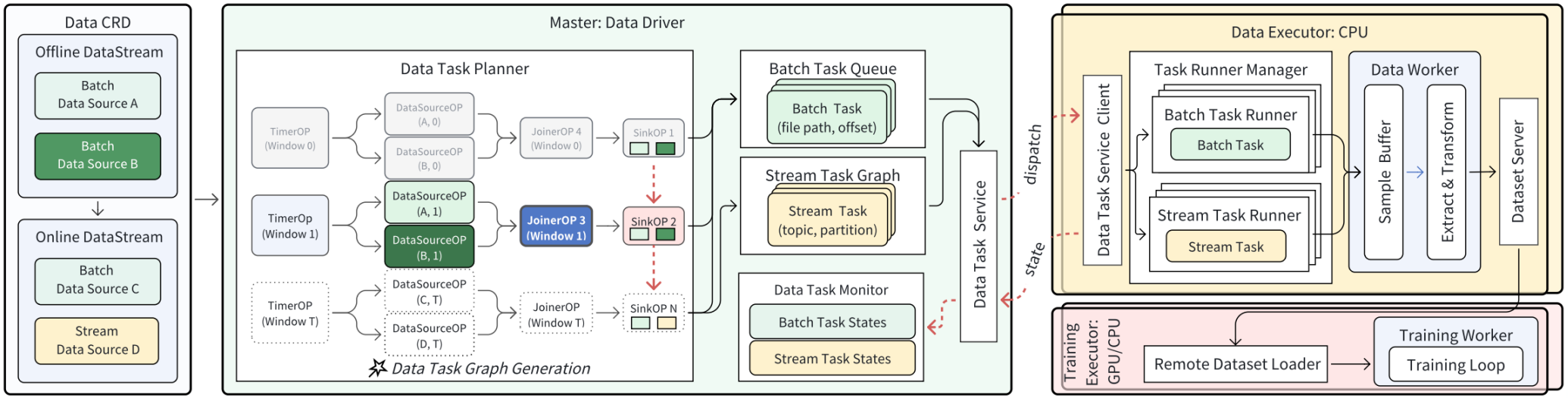
Contribute #1 Unified Resource Scheduling



- **Unified API-Server** upon diverse cluster
  - CRDs (Job CRDs *job customize resource definition,* Metric CRDs) for **standardize semantics**
- **Dynamic Scaling Manager**
  - Horizontal: efficient scaling executors for large clusters, auto-tuning in CPU/GPU collaboration
  - Vertical: fine-grained CPU/memory tuning per executor based on real-time metrics
  - **Combines horizontal and vertical scaling strategies**
- **Unified Resource Controller**
  - Standardizes YARN & Kubernetes **scheduling semantics**
  - Enables multi-cluster scheduling with automatic resource selection

ByteDance 字节跳动

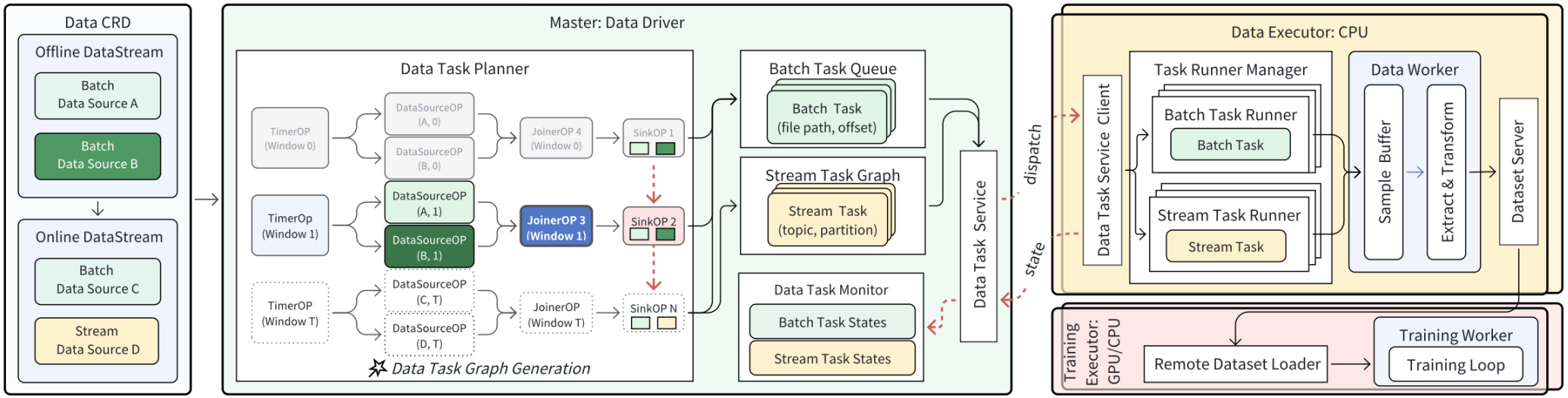
Contribute #2 Unified Data Orchestration



- **Three-Tier Data Definition**
  - Primus **Dataset**: Manages multiple data streams for a job
  - Primus **Data Stream**: Mixes & schedules data sources with sliding window,
  - Primus **Data Source**: a serial samples persisted upon different storage system & format
- **Data Task Graph Generation (DTGG)**
  - Four operator types: Timer OP, Data Source OP, Joiner OP, Sink OP
  - Generates data tasks across sources in **chronological order**

ByteDance 字节跳动

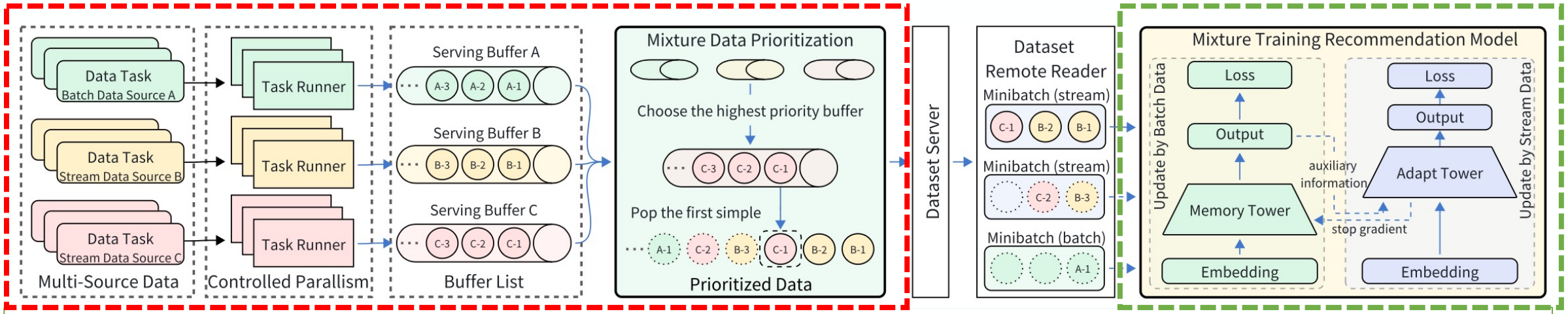
Contribute #2 Unified Data Orchestration



- **Data Driver:**
  - Generates tasks (DTGG)
  - Manages **batch task queues & stream graphs**
  - Orchestrates task assignments, task monitors, checkpoints
- **Data Executor:**
  - Supports batch & stream runners
  - Runs tasks & serves data (Apache Arrow)
  - Dynamic sharding, shuffling, and load balancing for **skew handling**
  - Support Remote CPU Offloading

ByteDance 字节跳动

Contribute #3 Unified Training Paradigm



- To effectively combine online stream data with offline batch data, leveraging the strengths of both online and offline training
  - **Mixture Training Recommendation Model (MTRM)**
    - **Controls the learning sequence of batch and stream data** through different neural network part for training
  - **Mixture Data Prioritization mechanism**
    - Optimizes the logic for multi-source data retrieval and **provides row-level priority control.**

ByteDance 字节跳动

Contribute #3 Unified Training Paradigm

- **Motivation**
  - Non-MTRM suffers from catastrophic forgetting & feedback delay
  - Batch data from different sources has varying output delays → need alignment with stream data
- **MTRM Framework**
  - **Memory Tower**: long-term parameter storage, processes historical batch data
  - **Adapt Tower**: focuses on recent stream data, adapts to latest trends
  - Embedding → tower forward → loss → tower-specific update

Algorithm 1: MTRM Training Process

Input: Continuous minibatches $\mathbf{X}$ from D_{pri}
Output: Updated MTRM with θ_M^{new} and θ_A^{new}

Initialize or load model parameters: memory tower θ_M, adapt tower θ_A;
Set learning rates α and loss functions $\mathcal{L}_M$, $\mathcal{L}_A$;
while *Not all minibatches* $\mathbf{X}$ *are processed* **do**
  if $\mathbf{X}$ *is* $\mathbf{X}_{batch}$ **then**
    $\mathbf{e}_{\text{batch}} \leftarrow \text{Embedding}(\mathbf{X}_{\text{batch}})$;
    $\mathbf{h}_M \leftarrow \text{MemoryTowerForward}(\mathbf{e}_{\text{batch}}, \theta_M)$;
    $\mathcal{L}_M \leftarrow \text{ComputeLoss}(\mathbf{X}_{\text{batch}}, \mathbf{h}_M, \theta_M)$;
    $\theta_M^{\text{new}} \leftarrow \theta_M - \alpha \nabla_\theta \mathcal{L}_M$;
  else if $\mathbf{X}$ *is* $\mathbf{X}_{stream}$ **then**
    $\mathbf{e}_{\text{stream}} \leftarrow \text{Embedding}(\mathbf{X}_{\text{stream}})$;
    $\mathbf{h}_{\text{aux}} \leftarrow \text{MemoryTowerForward}(\mathbf{e}_{\text{stream}}, \theta_M)$;
    $\mathbf{h}_A \leftarrow \text{AdaptTowerForward}(\mathbf{e}_{\text{stream}} \oplus \mathbf{h}_{\text{aux}}, \theta_A)$;
    $\mathcal{L}_A \leftarrow \text{ComputeLoss}(\mathbf{X}_{\text{stream}}, \mathbf{h}_A, \theta_A)$;
    $\theta_A^{\text{new}} \leftarrow \theta_A - \alpha \nabla_\theta \mathcal{L}_A$;
return θ_M^{new} or θ_A^{new}

ByteDance 字节跳动

Contribute #3 Unified Training Paradigm

- **Motivation**
  - Stream instances burst (e.g., hot events) → delay in stream data processing → model degradation
- **Prioritization Strategy**
  - Data workers compute priorities using buffer length & importance
  - Stream data assigned higher weights → processed first
  - Balances timeliness, backpressure, and resource limits

Algorithm 2: Prioritized Data from Serving Buffers

Input: Serving buffers $D = \{D_1, D_2, \ldots, D_n\}$, buffer weight $W = \{w_1, w_2, \ldots, w_n\}$, and buffer queued data length $Q = \{q_1, q_2, \ldots, q_n\}$

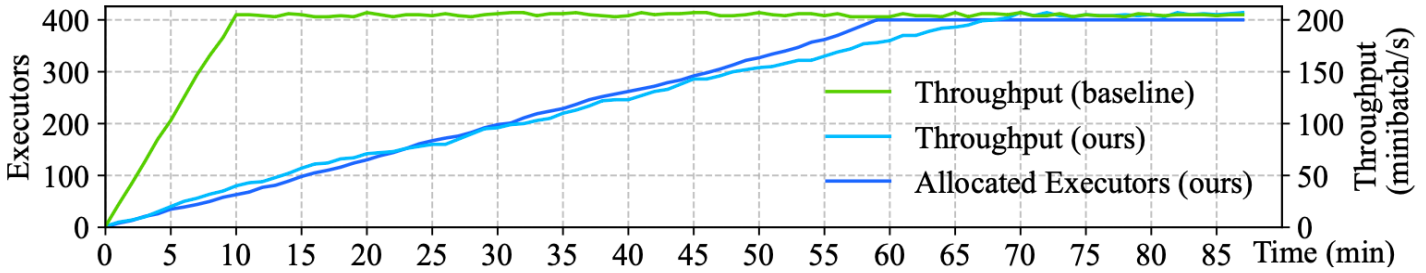
Output: Continuous prioritized data queue D_{pri}

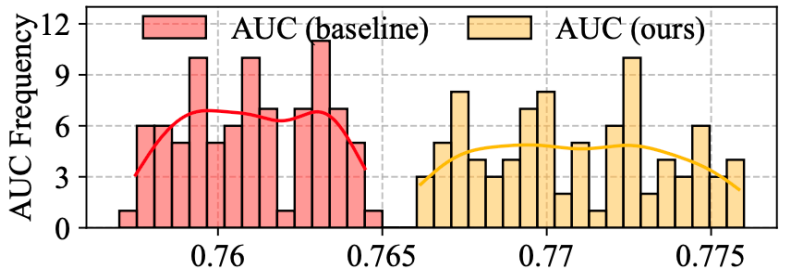
```
while Serving buffers D ≠ ∅ do
    maxIndex ← −1, maxPri ← −∞;
    for i ← 0 to bufferList.size − 1 do
        p_i ← w_i · sigmoid(q_i) ;
        if p_i > maxPri then
            maxIndex ← i, maxPri ← p_i ;
    sample ← D_maxIndex.pop();
    D_pri.append(sample);
return D_pri;
```

ByteDance字节跳动

Evaluations #1 Unified Resource Evaluation

- Dynamic Horizontal Scaling
  - Large-scale dynamic parallelism





Baseline: a fixed number of 400 data executors; Throughput: 205 → 210 minibatch/s, AUC: 0.4 ↑

  - Auto-tuning data & training executors ratio

Table 4: Dynamic Horizontal Scaling Comparision

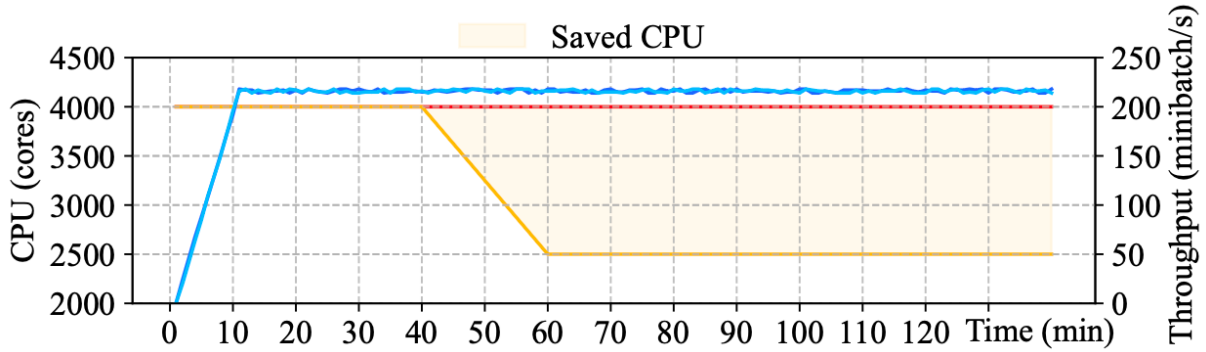
Method	CPU Used (cores)	AUC	Training Time (days)	Cluster ROI
Baseline	7200	0.9385	1.86	30.26
Ours	5520	0.9382	1.90	35.44+17.12%

Baseline: 900 8-core data & 8 training executors
Ours: 450 → 690 8-core data & 8 training executors
Job level: 23.33% saving in CPU
Cluster level: 17.12% increase in ROI

ByteDance 字节跳动

Evaluations #1 Unified Resource Evaluation

- Dynamic Vertical Scaling
  - Scenario 1: Excessive CPU resource allocation (400 10-core data executors)



Resource Efficiency

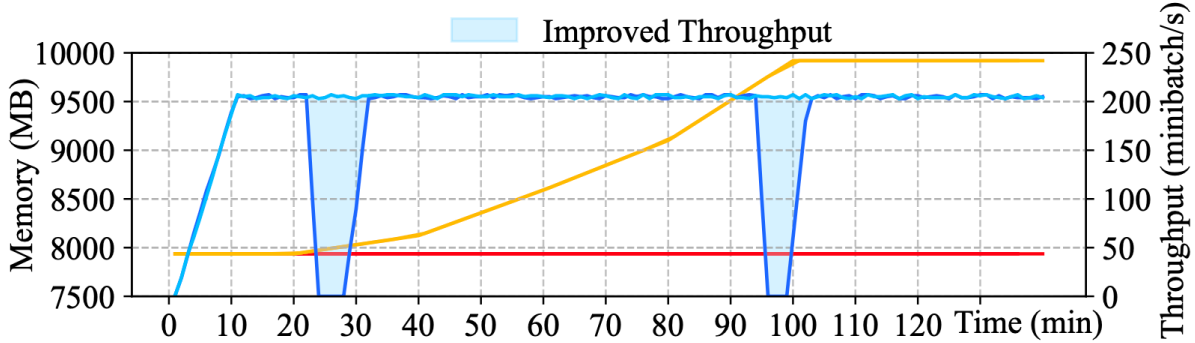
Scales down CPU cores

Total CPU: 4000 → 2400 cores

Per executor: 10 → ~6 cores

CPU utilization: 50% → 80%

  - Scenario 2: insufficient memory allocation (496 PS executors, each 16 GB memory)



Stable Training

Scales up memory

Total memory: 7936GB → 9920GB

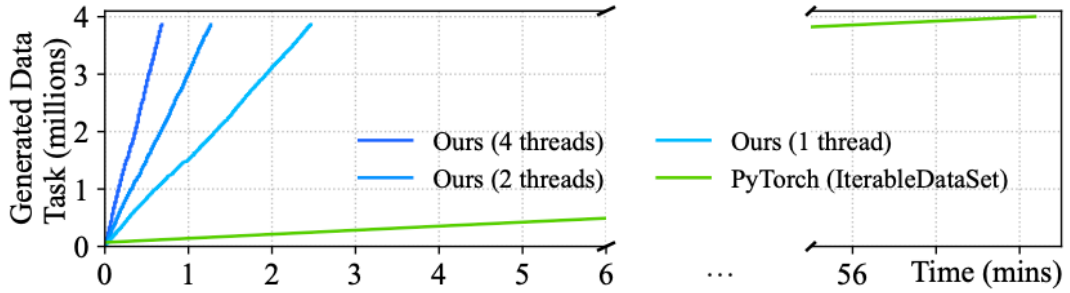
Training stuck time: 6.5% ↓

AUC: 0.62 → 0.78

ByteDance 字节跳动

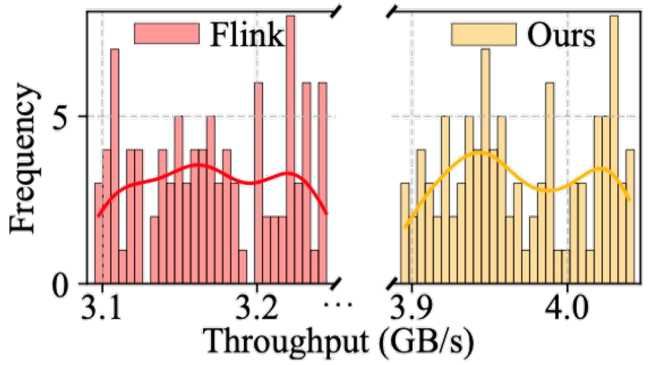
Evaluations #2 Unified Data Evaluation

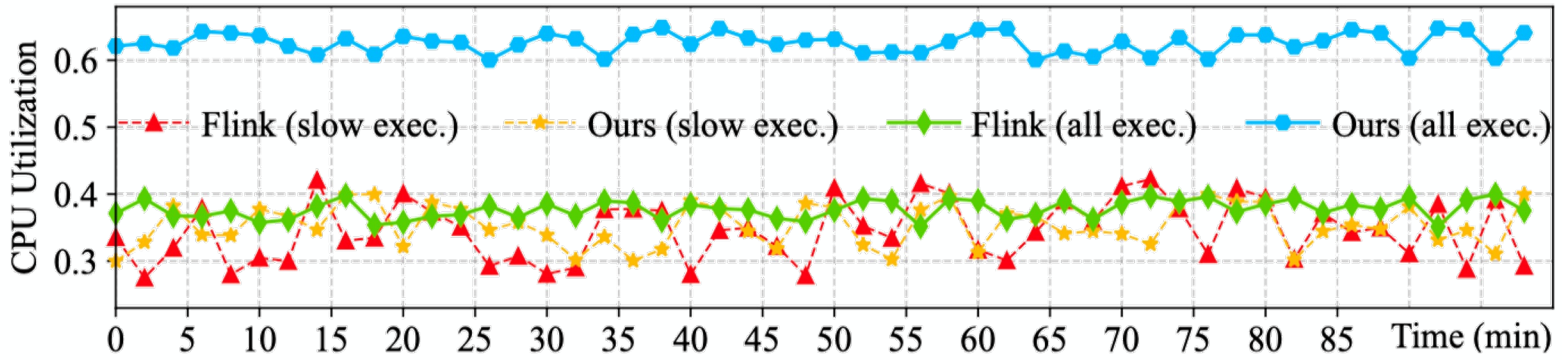
- Data Task Generation
  - A data driver (8 CPU cores, 25GB memory)
  - 20-day production data (two Feature Store tables, over 56 million underlying files



DTGG: 58 mins → 149s (1 thread, 23×) → 42s (4 thread)

- Steam Data Loading with All2All Shuffle
  - 40 executors (each 2 CPU cores, 4 GB memory), 10 are configured extremely slower





Flink vs Primus: Throughput 3.17 GB/s → 3.97GB/s 1.25×, CPU Utilization 37.6% → 62.7%

ByteDance 字节跳动

Evaluations #3 Unified Training Evaluation

- Mixture Training Recommendation Model
  - 4 production models for CVR and CTR
  - AUC improves 0.03% to 0.07%
  - Revenue increase 0.4%-2.4%

Table 5: Model and Resource Settings

No.	Model Basis*	Model Size (GB)	Executors Number	Executors CPU (cores)	Executors Memory (GB)
1	DSSM [24], LHUC [49]	600	60	13	45
2	FM [46], Transformer [51]	1200	40	6	30
3	FM, DIN [67], Transformer	800	50	12	35
4	EMSNet [20], LHUC	3700	50	8	30

*Production DLRMs absorb and integrate the various model structures listed.

Table 6: Comparison of AUC and Revenue Performance for Various Models without (w/o) and with (w/) MTRM

Model		Evaluation (AUC) Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Avg.	A/B Testing Samples	A/B Testing Revenue
1	w/o	0.92411	0.91992	0.92538	0.92737	0.92187	0.92554	0.92642	0.92437	2,252,037	-
	w/	0.92439+0.03%	0.92018+0.03%	0.92559+0.02%	0.92758+0.02%	0.92212+0.03%	0.92582+0.03%	0.92660+0.01%	0.92461+0.03%	2,252,459	+1.045%
2	w/o	0.87125	0.86748	0.87103	0.86939	0.86885	0.86855	0.86887	0.86935	21,643,587	-
	w/	0.87173+0.06%	0.86804+0.06%	0.87156+0.06%	0.86987+0.06%	0.86930+0.05%	0.86907+0.06%	0.86945+0.07%	0.86986+0.06%	21,643,223	+0.806%
3	w/o	0.95333	0.95083	0.95175	0.95098	0.95070	0.95218	0.95141	0.95160	21,141,647	-
	w/	0.95380+0.05%	0.95131+0.05%	0.95220+0.05%	0.95143+0.05%	0.95122+0.05%	0.95269+0.05%	0.95188+0.05%	0.95208+0.05%	21,148,298	+2.438%
4	w/o	0.83299	0.83337	0.83471	0.83443	0.83380	0.83460	0.83608	0.83428	254,869,721	-
	w/	0.83352+0.06%	0.83395+0.07%	0.83527+0.07%	0.83500+0.07%	0.83438+0.07%	0.83513+0.06%	0.83666+0.07%	0.83484+0.07%	254,864,629	+0.397%

ByteDance 字节跳动

Lessons: Learned in Primus Design & Implementation

- Road Towards Unified Resource Scheduling
  - Early hybrid deployment of YARN + Kubernetes (due to limited pod scheduling throughput & lack of gang semantics)
  - Primus abstracts scheduling differences, ensuring consistent user experience
  - Evolved to support dynamic horizontal & vertical scaling
- Optimizing Data Access for Various Formats
  - Challenges: TFRecord, ProtoBuf, 4mz + Snappy, ZSTD, GZIP
  - Solution: Arrow + NativeClient → 40% CPU reduction
  - Seamless integration of data lakes (Iceberg, Hudi) via ClassLoader isolation
- Necessity of Mixture Data Prioritization
  - Separate batch/stream executors → unified executors
  - Issue: delayed stream under peaks load, data leak issue
  - Solution: continuous prioritized data queue, ensuring stream data first

ByteDance字节跳动

Summary: 6-Year Development

- Primus: A unified training system deployed at ByteDance for **6 years**
- Unified Resource Scheduling: Multi-Strategies
  - Provides both dynamic horizontal and vertical scaling mechanisms
  - Distributed on clusters with 20M vCores and 200K GPUs
  - Increases CPU utilization from **50% to 80%**
- Unified Data Orchestration: DDTG
  - Supports batch and stream data mixture with a three-tier data definition
  - Provides a Graph Based Task Planner which accelerates training task generation by **23×**
- Unified Training Paradigm: MTRM
  - A new model with **memory and adaptive towers** to handle catastrophic forgetting and delayed feedback
  - Delivers **0.4%–2.4%** increase in advertising revenue

ByteDance字节跳动

THANKS

We are hiring

ByteDance 字节跳动

浙江大学 ZHEJIANG UNIVERSITY

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

Contact: shanjixi.inf@bytedance.com