Walle: An End-to-End, General-Purpose, and Large-Scale Production System for Device-Cloud Collaborative Machine Learning

Chengfei Lv (ZJU & Alibaba); Chaoyue Niu (SJTU & Alibaba); Renjie Gu, Xiaotang Jiang, Zhaode Wang, Bin Liu, Ziqi Wu, QiuLin Yao, Congyu Huang, Panos Huang, Tao Huang, Hui Shu, JinDe Song, Bin Zou, Peng Lan, Guohuan Xu (Alibaba); Fei Wu (ZJU); Shaojie Tang (UT Dallas); Fan Wu, Guihai Chen (SJTU)
Background & Motivation
Proliferation of Mobile Intelligent Services

Livestreaming  Speech Recognition  Recommendation
Bottlenecks of Cloud-Based ML Framework

**High Latency**
- Device-cloud interaction
- Process requests from millions or billions of users

**High Cost & Heavy Load**
- Communication & Storage
- Process data with complex ML algorithms

**High Privacy Risk**
- Upload sensitive raw data
- Store and process raw data on the cloud

Cloud takes all the load!

Mobile devices function only as user interfaces!

Upload Raw User Data

Return Results
Mobile devices and the cloud jointly accomplish ML tasks.

Need for Device-Cloud Collaborative ML

Overcome Cloud-Side Bottlenecks
- Reduce latency & communication cost
- Mitigate cloud-side load
- Keep private data on local devices

Natural Device-Side Advantages
- Close to users
- At data sources
Our Unique System-Level Consideration

| Application Layer |
|-------------------|
| Video Analytics (e.g., FilterForward in MLSys’19, Reducto & DDS in SIGCOMM’20), **Text Processing** (e.g., Gboard in MLSys’19), **Recommend** (e.g., DDCL in KDD’21, MPDA in KDD’22) |

Existing work was at the algorithm layer, normally for ML inference or training in a specific application.

| Algorithm Layer |
|-----------------|
| Device-Cloud Task Splitting Strategy (e.g., cloud training-device inference, Neurosurgeon in ASPLOS’17, federated learning in AISTATS’17), **Interaction Paradigm** (e.g., single device-cloud, multiple devices-cloud), **Collaboration Mechanism** (e.g., through exchanging data or model) |

| System Layer |
|--------------|
| **How to build a general-purpose system that can put device-cloud collaborative ML in large-scale production?** |

| Hardware Layer |
|----------------|
| (Mobile Devices & Cloud Servers) |
2

Overall Goal & Architecture
Walle – Overall Goal

End-to-End
- Develop, deploy, runtime
- All three phases of ML task
- Both sides of device and cloud

Heterogeneous hardware & software of mobile devices & cloud servers

Hundreds of CV, NLP, recommendation tasks in large-scale production
Walle – Overall Architecture

Oriented by ML task
- **Scripts** (e.g., Python codes for three phases of ML task)
- **Resources** (e.g., data, models, dependent libraries)
- **Configurations** (e.g., trigger conditions)

### Deployment Platform
- Task Management
- Task Release & Deployment

### Compute Container
- Standard APIs
- Python Thread-Level VM
- Data & Model Related Libraries
- Tensor Compute Engine
- Backends (Device & Cloud)

### Data Pipeline
- Device-Cloud Tunnel
- On-Device Stream
- Processing Framework
- User Behavior Data

ML task execution
ML task input preparation
ML task management & deployment
3

Walle – Compute Container
Practical Challenges

**Monthly/Weekly APP update vs. Daily ML task iteration**

**Heterogeneous hardware & software of mobile devices & cloud servers**

**Diverse CV, NLP, and recommendation tasks**

**Resource limitation of a certain mobile APP**
### Architecture

#### Python
- dynamically-typed
- widely used

#### C/C++
- cross-platform
- high-performance

---

### First!

#### ML Task Scripts
- decouple daily task iteration from weekly/monthly mobile APP update

#### Standard APIs

#### Python Thread-Level VM

#### Data Processing & Model Execution Libraries
- MNN-Matrix
- MNN-CV
- MNN-Inference
- MNN-Training

#### Tensor Compute Engine
- Geometric Computing
- Semi-Auto Search

#### Backends
- ARM v7/v8/v8.2
- OpenCL/Vulkan/Metal
- x86 AVX/AVX-512
- CUDA

#### OS & Hardware
- Android/iOS
- Linux/Windows/MacOS/Docker
- CPU/GPU/NPU

---

### Integrated Design

- **Exposé** high performance of tensor compute engine
- **Reduce** the workload of optimizing each library for heterogeneous backends
- **Support** the whole cycle of ML tasks
- **Keep** package small

### Open Source

- [https://github.com/alibaba/MNN](https://github.com/alibaba/MNN)
- [https://www.mnn.zone/](https://www.mnn.zone/)

- 6.8k stars
- 1.4k forks
Tensor Compute Engine – Design Principle

**Manual Operator Optimization**

- Composite Ops (16)
- Control-Flow Ops (2)
- Transform Ops (45)
- Atomic Raster Op (1)
- Atomic Ops (61)
- Hardware Backends (16)

**Graph-Level Runtime Optimization**

- a series of operators: $op_1 \rightarrow op_2 \rightarrow \cdots \rightarrow op_n$
- available backends: $ba_1, ba_2, ba_3$

**Backends**

| Backends   | Algorithm | SIMD | Memory | Assembly |
|------------|-----------|------|--------|----------|
| ARM (Device) | ✓        | ✓    | ✓      | ✓        |
| GPU (Device) | ✓        | ✓    | ✓      | X        |
| x86 (Server)  | ✓        | ✓    | ✓      | ✓        |
| CUDA (Server)  | X        | ✓    | ✓      | X        |

**Search Strategies**

- Dynamic Deployment
- Light Workload
- Manual Experience
- OPT

| Search Strategies | Dynamic Deployment | Light Workload | Manual Experience | OPT |
|-------------------|--------------------|----------------|--------------------|-----|
| Manual            | ✓                  | X              | ✓                  | X   |
| Auto (TVM)        | X                  | ✓              | X                  | ✓   |
| Semi-Auto         | ✓                  | ✓              | ✓                  | ✓   |

reduce roughly 46% workload

quickly find min-cost backend
Python Virtual Machine (VM) – Refining CPython

Package Tailoring for APP Need

Functionality Tailoring
- Keep only interpreter for mobile devices

Library & Module Tailoring
- Keep only 36 necessary libraries (e.g., abc, type, re, functools, etc)
- Keep only 32 necessary modules (e.g., zipimport, sys, exceptions, gc, etc)

10MB+ to 1.3MB (ARM64-based iOS)

First in industry to be ported to mobile devices!

Task-Level Multi-Threading

Motivations
- The global interpreter lock (GIL) & Single process of mobile APP → parallel \( X \)
- Practical characteristics of ML tasks
  - Concurrent triggering of many tasks
  - Independence across different tasks
  - Sequential execution of different phases in each individual task

How?
- Bind each ML task with a thread
- Do thread isolation

Abandon GIL and support multi-threading!
Walle – Data Pipeline
Bottlenecks of Mainstream Data Pipeline

Cloud-Based Stream Processing

Raw Data of Massive Users

Flink

Features

Transfer 500KB Data in 5min
(modified http request)

Offline Tunnel

Basic Events

- page enter
- page scroll
- exposure
- click
- page exit

Process User Data Far Away from Source

- Device-cloud communication for redundant raw data
- Cloud-side computation & storage for aggregate data from billions of users

- Time-Consuming
- Resource-Consuming
- Error-Prone
- Privacy-Sensitive
Enable each mobile device to process only its user’s behavior data at source
5

Walle – Deployment Platform
Practical Considerations & Challenges

Frequent experiment & deployment for daily ML task iteration

Massive multi-granularity task deployment requirements

Intermittent device availability

Potential task failure

- Unstable wireless network
- Frequent APP switch & Only one APP on the foreground
- Each mobile APP has one process. Failure of any task leads to APP crash.
- Extensive real-device testing is impractical.

- Hundreds
- Billion

- • APP version
- • Device-side differentiation
- • User-side differentiation
Timely, Robust Task Release & Deployment

Cloud-based simulators with compute container

Deployment Strategy
- Uniform Deployment
- Coarse-Grained Grouping
- Mobile App Version
- Shared Resources
- Customized Deployment
- Fine-Grained Grouping
- Device-Side Information
- User-Side Information
- Exclusive Resources

Release
- Test
- Rollback
- Beta
- Gray Release
- Monitor
- Exception Statistics

Real-Time Reach
- CDN
- Push Service
- CEN
- Pull Service

Device
- Decode
- Query
- Store
- Execute
- Monitor

Existing client-side http request for business services
6

Evaluation Results
6.1 Practical Performance in E-Commerce Scenarios
Compute Container in Livestreaming

Cloud-Based Design
- Key bottleneck: **Heavy load** (lots of streamers, long video streams, stringent latency requirement)
- Cover **part** of streamers
- Analyze **part** of video frames

Device-Cloud Co-Design
- Cloud-side load: **-87%**
- #Covered streamers: **+123%**
- #Daily recognized highlights per unit of cloud cost: **+74%**
- Overall latency per highlight recognition: **< 150ms**

| Model          | Parameter Size | Item Detection | Item Recognition | Facial Detection | Voice Detection |
|----------------|----------------|----------------|------------------|-----------------|----------------|
| Huawei P50 Pro | 56.92ms        | 25.68ms        | 41.42ms          | 0.07ms          |
| iPhone 11      | 33.71ms        | 29.74ms        | 22.58ms          | 0.01ms          |

Roughly 12% of highlights recognized with low confidences on mobile devices need to be processed by cloud-based big model.
Data Pipeline in Recommendation

Cloud-Based Data Pipeline
- **Time-Consuming:** 33.73s per IPV feature generation (using Alibaba’s Blink)
- **Resource-Consuming:** 253.25 CU (1 CU denotes 1 CPU core + 4GB memory)
- **Error-Prone:** 0.7% error rate

Walle’s New Data Pipeline
- **Lower On-Device Latency:** 44.16ms per IPV feature generation
- **Lower Communication & Storage Cost**

| Raw Events | IPV Feature | IPV Encoding |
|------------|-------------|--------------|
| Size       | Reduction   | %            |
| 21.2KB     | /           | 93.9%        |
| 1.3KB      | 99.4%       |              |
| 128B       |             |              |

- **No Feature Error**
ML Task Deployment Statistics

Large-Scale Production Use

- As part of Alibaba’s ML backbone infrastructure
- Put in use since 2017 & already run for roughly **1,500 days**
- Invoked **153 billion+** times per day
- Deployed **1,000+** kinds of ML tasks in total, each with **7.2** versions on average
- Supporting **30+** mobile APPs
- Supporting **300+** kinds of active ML tasks for **0.3 billion** daily active users with mobile devices

Cover all 7 million online devices in 7min and all the target 22 million devices in 19min
6.2
Extensive Micro-Benchmark Testing Results
MNN vs. TensorFlow (Lite) & PyTorch (Mobile)

MNN outperforms other frameworks in almost all the test cases and is more full-featured on the side of mobile devices.
**MNN vs. TVM**

- **TVM** (autotuning + compiling) roughly costs **thousands of seconds**. **MNN’s semi-auto search** for runtime optimization costs roughly **hundreds of milliseconds**.

- **MNN** can support the industrial scenarios that involve numerous heterogeneous devices and require frequent and quick task iteration, whereas **TVM cannot**.

---

**MNN outperforms TVM due to manual operator optimization.**
Python Thread-Level VM, Real-Time Tunnel

Python Thread-Level VM vs. CPython with GIL (analyzed over 30 million online ML task executions)

Performance Improvement

| Task Category       | Performance Improvement |
|---------------------|-------------------------|
| Light-Weight        | 52.11%                  |
| Middle-Weight       | 144.36%                 |
| Heavy-Weight        | 25.70%                  |
|                      | Light-Weight [0, 100) ms|
|                      | Middle-Weight [100, 500) ms|
|                      | Heavy-Weight [500, 1200) ms|

Task-level multi-threading without GIL is the key of performance boosting.

Practical Delay of Real-Time Tunnel with Varying Size of Data Upload (analyzed over 364 million uploads)

90% uploads < 3KB, 250ms
0.1% uploads = 30KB, 450ms
7

Summary
Design and build the first end-to-end, general-purpose, and large-scale production system, called Walle, for device-cloud collaborative ML. Compute container comprises MNN, which introduces geometric computing to sharply reduce the workload of manual operator optimization, and semi-auto search to identify the best backend with runtime optimization; and a Python VM, which abandons GIL and supports task-level multi-threading, and also is the first to be ported to mobile devices. Data pipeline introduces on-device stream processing with trie-based concurrent task triggering to enable processing user behavior data at source. Deployment platform supports fine-grained task release and deployment to billion-scale devices with strong timeliness and robustness. Evaluation in practical e-commerce scenarios and extensive micro-benchmarks have demonstrated the superiority of Walle. Walle has been in large-scale production use in Alibaba, while MNN has been open source with a broad impact in the community.
Thanks for listening!
Comments & Questions?