KAITIAN: Communication for Heterogeneous Accelerators in Embodied AI

Jieke Lin*†, Wanyu Wang*†, Longxiang Yin†, Yinhe Han†

*University of Chinese Academy of Sciences, Beijing, China
Email: {linjinke23, wangwanyu23}@mails.ucas.ac.cn
†Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
Email: {yinlongxiang, yinhes}@ict.ac.cn

***Abstract*—Embodied AI systems demand real-time, energy-efficient computation, often relying on heterogeneous accelerators such as GPGPUs and NPUs. However, the use of proprietary communication libraries introduces interoperability barriers, limiting collaboration and resource utilization. We present KAITIAN, a distributed communication framework that integrates vendor-specific libraries with general-purpose communication layers, combined with a load-adaptive scheduling mechanism for efficient task allocation. Implemented within PyTorch and evaluated on NVIDIA GPUs and Cambricon MLUs, KAITIAN improves resource utilization, enables scalable distributed training, and accelerates training time by up to 42% with minimal communication overhead.**

I. Introduction

Embodied intelligent systems, such as autonomous robots, operate in dynamic environments, requiring real-time perception, decision-making, and control [1]. To meet these demands, they increasingly integrate heterogeneous accelerators—GPGPUs for parallel computing, NPUs for deep learning, and domain-specific accelerators for specialized tasks. However, proprietary communication libraries across different vendors are typically incompatible, hindering collaborative computation across heterogeneous devices. This leads to resource underutilization and communication bottlenecks during distributed training.

To address these challenges, we propose **KAITIAN**, a distributed communication framework that seamlessly integrates proprietary and general-purpose libraries, coupled with a load-balancing mechanism to optimize task allocation. By enabling efficient cross-device communication, KAITIAN effectively harnesses heterogeneous computing resources for embodied AI applications.

II. KAITIAN Framework Design

A key obstacle in heterogeneous computing is that mainstream deep learning frameworks, such as PyTorch [2], are designed around a single communication backend, limiting their ability to natively orchestrate multiple vendor-specific libraries within the same training job. This constraint hampers the collaborative use of heterogeneous accelerators.

KAITIAN addresses this challenge through a pluggable extension to PyTorch that seamlessly integrates diverse communication backends into a unified framework. Its design follows two key principles:

- **Intra-Group Communication**: Within homogeneous accelerator groups (e.g., GPUs or MLUs), KAITIAN leverages vendor-optimized libraries (such as NCCL and CNCL) to maximize communication efficiency.
- **Inter-Group Communication**: Between heterogeneous groups, KAITIAN relays data through host CPU memory using Gloo, a general-purpose communication layer, to enable cross-accelerator collaboration.

A. Hybrid Communication Architecture

As illustrated in Figure 1, KAITIAN organizes processes into *containers* based on device type. It then applies a hybrid strategy:

Intra-Group Communication employs fast, device-specific libraries for local synchronization within each container, achieving near-native performance. In contrast, **Inter-Group Communication** routes tensors through CPU host memory: data is copied from the source accelerator to host memory, exchanged across containers via Gloo, and transferred to the target accelerator. Although this relay incurs additional memory copies, it enables otherwise incompatible devices to collaborate within a unified training workflow.

By decoupling intra-group efficiency from inter-group interoperability, KAITIAN enables scalable distributed training across diverse hardware, without sacrificing communication performance within homogeneous subgroups.

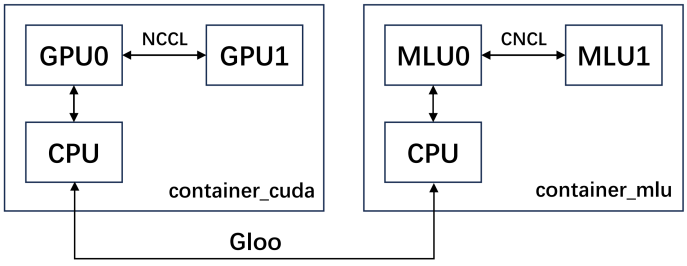

Fig. 1. Architecture of the KAITIAN Hybrid Communication Framework.

B. Load-Adaptive Mechanism

Heterogeneous accelerators often show performance disparities, limiting system efficiency in synchronous tasks like data-parallel training. KAITIAN's load-adaptive mechanism addresses this:

- **Benchmarking**: A program evaluates each accelerator's performance before tasks. The fastest device's runtime sets the reference (score = 1); others are scored as $\text{score}_i = \text{time}_{\min}/\text{time}_i$.
- **Dynamic Allocation**: We define data allocation as: $\text{batch\_size}_i = \text{original\_batch\_size} \times \text{score}_i$, which adjusts based on scores, ensuring uniform computation times across devices.

This balances loads, maximizing throughput for robotic applications needing stable performance.

III. Implementation and Evaluation

A. Implementation Details

We implement KAITIAN as a extension PyTorch communication backend, overriding DistributedSampler for load adaptation. Command-line tools manage tasks. Docker ensures environment isolation, with Redis for discovery and state synchronization.

B. Experimental Setup

- **Hardware**: A heterogeneous server equipped with 2 NVIDIA GTX 1080 GPUs, 2 Cambricon MLU370-S4 accelerators, an AMD EPYC 7763 CPU, and 64 GB of RAM.
- **Software**: Ubuntu 20.04, Docker, NVIDIA Driver and CUDA Toolkit, Cambricon Driver and CNToolkit.
- **Task**: Image classification on CIFAR-10 [3] using MobileNetV2.

C. Evaluation Results

We evaluate KAITIAN under the experimental setup described above, using NCCL and CNCL as baseline systems. The evaluation focuses on training time and model accuracy.

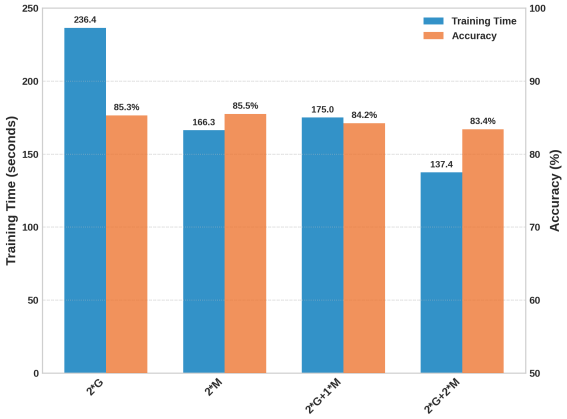

Fig. 2. KAITIAN Training Efficiency and Accuracy Comparison.

Figure 2 compares training time and accuracy across different accelerator configurations. KAITIAN, using 2 GPUs and 2 MLUs (2G+2M), achieved the fastest training time at 137.4 seconds, representing a 42% improvement over the GPU-only NCCL baseline (236.4 seconds) and a 17% improvement over the MLU-only CNCL baseline (166.3 seconds). Accuracy remained comparable across all configurations, with 83.4% for 2G+2M versus 85.3% for 2G and 85.5% for 2M, indicating that KAITIAN's heterogeneous communication introduces minimal accuracy degradation.

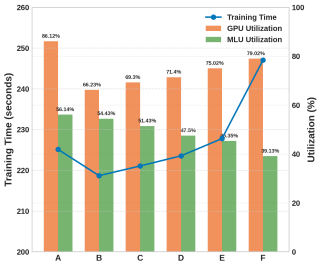

Fig. 3. Load Adaptive Mechanism.

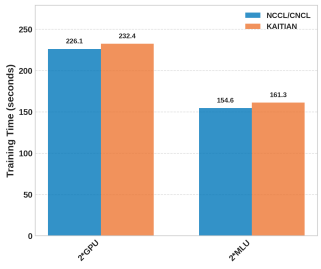

Fig. 4. Communication Overhead.

Based on the dynamic load adjustment method introduced in the Load-Adaptive Mechanism section, we evaluate the effect of load balancing strategies on heterogeneous training efficiency, as shown in Figure 3. Different device utilization levels significantly impact system performance. By dynamically adjusting the load during training, KAITIAN is able to converge to the optimal strategy B, achieving balanced GPU and MLU utilizations and minimizing overall training time.

Figure 4 presents the communication overhead of KAITIAN compared to native libraries in homogeneous environments. KAITIAN introduced minimal overhead, with a training time of 232.4 seconds versus 226.1 seconds for NCCL on 2 GPUs (2.8%), and 161.3 seconds versus 154.6 seconds for CNCL on 2 MLUs (4.3%). These results demonstrate the efficiency of KAITIAN's inter-group communication strategy.

IV. Conclusion

KAITIAN enables efficient heterogeneous computing for embodied AI by integrating specialized and general-purpose communication libraries with a load-adaptive mechanism. It maximizes resource utilization, supports complex model deployment, and accelerates algorithm development on diverse accelerators like GPUs and MLUs. Evaluations show up to 42% faster training with minimal overhead and strong scalability. Applicable to edge robotic platforms, KAITIAN's future includes porting to robotic hardware and benchmarking on tasks like SLAM and grasping. Source code is available at https://github.com/jklincn/kaitian.

References

[1] W. Sun, S. Hou, Z. Wang, B. Yu, S. Liu, X. Yang, S. Liang, Y. Gan, and Y. Han, "DaDu-E: Rethinking the Role of Large Language Model in Robotic Computing Pipeline," *arXiv preprint arXiv:2412.01663*, 2024.

[2] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Vancouver, BC, Canada, Dec. 2019, pp. 8026–8037. [Online]. Available: https://papers.nipsskapa.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

[3] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," Toronto, ON, Canada: Univ. of Toronto, Tech. Rep., Apr. 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.