Demo: Real-Time Inference on GPU-based Heterogeneous SoCs with GPU Cache Locking

Kehao Ma
Shandong University
Qingdao, China

Wei Zhang
Shandong University
Qingdao, China

Mengying Zhao
Shandong University
Qingdao, China

Lei Ju
Shandong University
Qingdao, China

***Abstract*—Embedded real-time systems are increasingly turning to GPU-based SoCs to efficiently handle machine learning tasks at the edge. Modern GPU SoCs often feature specialized AI accelerators to enable concurrent CNN inference while maintaining energy efficiency. While this trend typically leads to a general improvement in performance, the integration of multiple AI accelerators presents challenges for building real-time systems, where ensuring timing predictability is a key design goal. On one hand, contention among various computation units over shared memory can introduce non-deterministic timing behaviors. On the other hand, the built-in GPU scheduling mechanism cannot assure that emergent tasks to be executed first, potentially violating real-time constraints. To tackle these challenges, this study introduces a timing predictability-aware cache locking policy to reduce main memory access volume and mitigate main memory contention with AI accelerators. Additionally, a real-time scheduling framework is proposed to bypass the inherent GPU scheduling algorithm.**

***Index Terms*—Real-time, GPU, Heterogeneous SoCs**

I. Introduction

Deep neural networks (DNNs) are now being widely implemented on embedded systems to power intelligent edge computing. In order to enable real-time execution of computational intensive DNN tasks at the edge, the adoption of GPU-based Systems-on-Chip (SoCs) has become prevalent in embedded systems. To keep up with the growing performance requirements, modern SoCs are increasingly employing heterogeneous architectures that combine general-purpose GPUs with specialized AI accelerators. For example, notable commercial-off-the-shelf (COTS) devices like the NVIDIA Orin SoC [1] integrate two dedicated Deep Learning Accelerators (DLAs) alongside.

While the integration of AI accelerators significantly boosts computational throughput and energy efficiency, it presents critical challenges for real-time systems. DNN tasks typically maintain a consistent workload across various inputs, demonstrating deterministic timing behavior when executed individually. However, when other tasks run concurrently on AI accelerators, contentions over shared memory between the GPU core and AI accelerators can introduce non-deterministic memory access latency, resulting in unpredictable timing behaviors. Our experiments indicate that the ratio between the worst-case execution time and the best-case execution time increases from 1.04 to 1.47 when other tasks are simultaneously executed on AI accelerators. Real-time systems must ensure that all tasks are completed before their deadlines. This non-deterministic timing behavior makes the precise and secure prediction of the worst-case execution time of DNN tasks challenging, increasing the risk of timing constraint violations.

Moreover, contemporary GPU SoC scheduling policies are not publicly available and are designed to boost system throughput [2], which conflicts with the goal of real-time systems to minimize the response time of high-priority tasks to avoid missing deadlines. This disparity presents a challenge when attempting to schedule real-time tasks using the existing built-in task scheduling policies. Specifically, GPU SoCs employ a blocking synchronization mechanism where the host halts to await the completion of DNN tasks on the GPU core and AI accelerators. As a result, incoming high-priority tasks are required to wait for all ongoing tasks to be completed, resulting in a significant degradation in response time.

To address the above-mentioned issues, we first utilize the cache locking mechanism to store data in the cache, reducing the volume of main memory access, as well as the main memory contentions. Given the limited cache capacity (e.g., NVIDIA Orin only provides 3 MB for locking), the selection of data to lock is pivotal in effectively managing main memory access [3]. In our paper, we introduce a reinforcement learning-based approach to determine the optimal data for minimizing main memory access volume. Additionally, we present a real-time scheduling framework that combines a software-defined scheduling component and a non-blocking synchronization mechanism. The real-time scheduling framework empowers users to define task scheduling policies without being constrained by the inherent GPU scheduling mechanism.

II. Methodology

The system's overview architecture is depicted in Figure 1, comprising a cache locking agent and a real-time task management component.

A. Reinforcement learning-based cache locking method

During DNN execution, two primary types of data are accessed: weights and activations. Weights are specific to each layer and are not shared across layers, whereas activations are commonly stored in a reusable activation buffer during inference and are reused across different layers. Given this, our approach focuses on locking activations to decrease the main memory access volume.

2325-1301/25/$31.00 ©2025 IEEE
DOI 10.1109/RTCSA66114.2025.00039

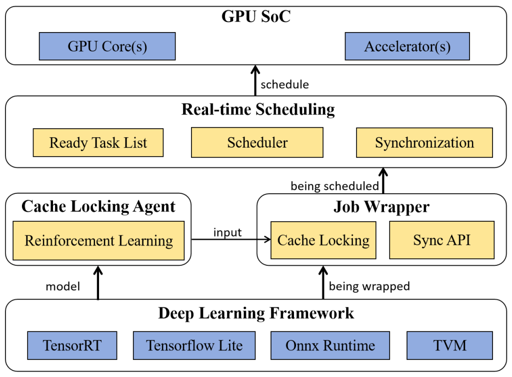

Fig. 1. Overview of the propose system.

We develop a Proximal Policy Optimization (PPO)-based reinforcement learning method to optimize the timing predictability of neural networks with cache locking. In this method, the environment is defined by the computational platform and target network, which takes locking policies as input and offers the ratio between the worst-case execution time and the best-case execution time as a stable proxy metric for enhancing stability. The state is encoded as an n-dimensional binary vector that indicates locked or unlocked blocks in the activation buffer. Actions are responsible for selecting specific blocks to lock (ranging from 0 to n-1) or for executing deliberate null operations (action n) when locking no longer provides substantial benefits. Our agent utilizes dual actor-critic networks featuring fully-connected layers and ReLU activations. The actor component generates action probabilities using softmax, while the critic estimates state-action values. The reward function gauges performance improvement by measuring the reduction in the ratio between the worst-case execution time and the best-case execution time compared to the previous policy. Through iterative training, this methodology converges towards an optimized locking configuration that minimizes latency while enhancing stability by selectively locking activation buffer blocks.

B. Real-Time scheduling

The proposed Real-Time scheduling framework introduces a list to buffer tasks before dispatching them to the GPU cores or AI accelerators. This list allows user-defined scheduling policies that enforce timing constraints through dynamic execution ordering. Furthermore, we introduce a CPU-GPU synchronization mechanism that notifies the CPU when GPU kernel execution is immediately finished, to shrink the CPU stalls during the GPU kernel scheduling. This is achieved through a kernel wrapper that writes completion signals to shared memory once the GPU kernel is finished, coupled with an interrupt-driven polling system where the CPU efficiently checks signals without busy-waiting.

III. Evaluation

We conducted experiments on Nvidia Jetson Agx Orin with an Ampere architecture GPU and two Deep Learning

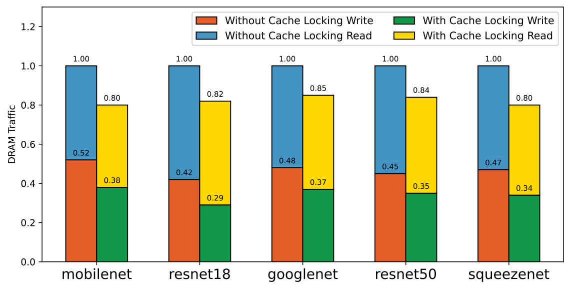

Fig. 2. DRAM traffic under different settings.

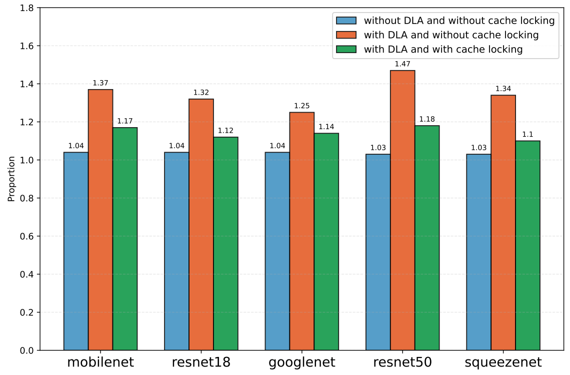

Fig. 3. Ratio between the worst-case execution time and the best-case execution time under different settings.

Accelerators (DLAs), running on JetPack 5.1. The models used in our experiments are sourced from ONNX Model Zoo [4], and we employed TensorRT 8.5 to compile them into inference engines. Experimental results illustrate that, owing to the cache locking mechanism, main memory traffic is reduced by an average of 20% as shown in Figure 2. Consequently, the ratio between the worst-case execution time and the best-case execution time is decreased by up to 24% as shown in Figure 3, demonstrating a relatively predictable timing behavior.

Furthermore, we evaluate the proposed real-time scheduling framework by comparing it to the built-in GPU hardware scheduler, which executes different tasks concurrently in an interleaved manner. We assume that 10 concurrent ResNet tasks are released simultaneously, each with different assigned priorities. Experimental results show that our scheduling framework reduces the response time of the top 30% of high-priority tasks by 58%, while also decreasing the average response time for all tasks by 23%.

References

[1] L. S. Karumbunathan, "Nvidia jetson agx orin series," *Online at https://www. nvidia. com/content/dam/en-zz/Solutions/gtcf21/jetson-orin/nvidia-jetson-agx-orin-technical-brief. pdf*, 2022.
[2] K. K. Ng, H. M. Demoulin, and V. Liu, "Paella: Low-latency model serving with software-defined gpu scheduling," in *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 595–610, 2023.
[3] A. M. González, J.-B. Chaudron, R. Leconte, Y. Bouchebaba, and D. Doose, "Exploring igpu memory interference response to l2 cache locking," in *21st International Workshop on Worst-Case Execution Time Analysis*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.
[4] "Onnx Model Zoo." https://github.com/onnx/models, 2025.