RAPID-ASR: Runtime ASR Parallelization and Isolation on Mobile Devices

Chong Tang
University of Southampton
chong.tang@soton.ac.uk

Hao Dai
University of Southampton
h.dai@soton.ac.uk

Jagmohan Chuahan
University College London
jagmohan.chauhan@ucl.ac.uk

I. Introduction

On-device ASR is critical for privacy and real-time applications such as voice assistants and live translation. Frameworks like TensorFlow Lite optimize static models via quantization and operator fusion [1], but problems arise when using the dynamic control flow and variable-length tensor of transformer ASRs[2], which support multi-head attention and token prediction [3]. Consequently, CPUs remain the most practical inference engine, but their parallelism ability is overlooked. RAPID-ASR bridges this gap by uniting runtime graph partitioning, branch-aware memory allocation, and multithreaded execution.

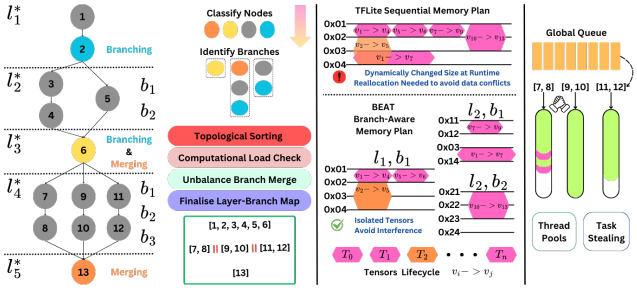

Fig. 1: System Overview

II. Method

a) Method Explanation: RAPID-ASR begins by performing *runtime graph analysis* on the transformer's dynamic compute graph at inference time, partitioning nodes into dependency-resolved layers and isolating independent branches for potential parallel execution; it then employs *branch-aware memory allocation*, tagging each tensor with a combined Layer–Branch–Index identifier and placing it in a dedicated memory arena to eliminate read-after-write conflicts and manage lifecycles (including safe dynamic reallocations) at branch granularity; finally, BEAT's *multithreaded execution* dispatches these branch tasks to a fixed pool of worker threads with dynamic work-stealing, minimizing synchronization to layer boundaries and maximizing CPU core utilization without any intrusive model modifications (Fig. 1).

b) Evaluation Results: To assess RAPID-ASR's effectiveness, we deployed Whisper and Conformer-CTC ASR models [4] via TensorFlow Lite 2.17.0 on four representative CPU platforms (Dimensity 8100, Kirin 980, Google Tensor, and Raspberry Pi 4B). Using LibriSpeech test clips (3–15 s at 16 kHz), we measured end-to-end inference latency, memory allocation performance (including peak arena usage), and total energy consumption, averaging five runs per configuration on all available CPU cores. Compared to the standard TFLite CPU runtime, RAPID-ASR achieves up to 38.5 % reduction in latency (especially on models with extensive branching), accelerates memory allocation by up to 15.6× (Fig. 2) while keeping peak memory overhead below 5 MB, and reduces total energy consumption by up to 24.6 % through higher instantaneous CPU utilization over shorter runtimes.

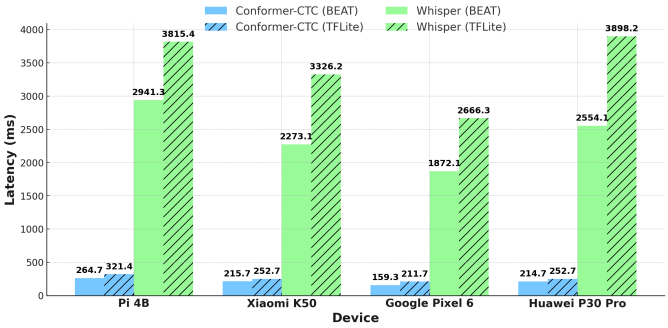

Fig. 2: End-to-end inference latency comparison of Whisper and Conformer.

III. Conclusion

RAPID-ASR is a lightweight transformer ASR runtime for mobile CPUs that uses branch-aware graph analysis and memory isolation to deliver significant reductions in latency, memory footprint, and energy consumption. Future work will add adaptive scheduling for shared-resource environments and dynamic-tensor support on NPUs, further closing the gap to heterogeneous edge inference.

References

[1] M. Abadi, A. Agarwal, and et al., "Tensorflow: Large-scale machine learning on heterogeneous systems," Software available from tensorflow.org, 2015.

[2] A. Vaswani, N. Shazeer, and et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, 2017, vol. 30.

[3] X. Yang, Q. Qi, and et al., "Towards efficient inference: Adaptively cooperate in heterogeneous IoT edge cluster," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2021, pp. 12–23.

[4] A. Gulati, J. Qin, and et al., "Conformer: Convolution-augmented Transformer for Speech Recognition," *arXiv preprint arXiv:2005.08100*, 2020.