Accelerating AI Models with Andes Matrix Multiplication (AMM) from CNN to LLM

Pei-Hsiang Hung, Chung-Hua Yen, I-Wei Wu
Andes Technology

Abstract

Matrix multiplication is a fundamental operation in AI, image processing, and scientific computing, requiring efficient hardware acceleration to meet performance demands. To address this challenge, RISC-V International (RVI) formed the Integrated Matrix Extension (IME) Task Group. To serve immediate market demands, Andes Technology implements the Andes Matrix Multiplication (AMM) instruction set, an extension to the RISC-V Vector Extension (RVV), which Andes presented in IME meetings. This talk presents our integration of AMM into the Intermediate Representation Execution Environment (IREE) for optimized execution on AMM-enabled RISC-V systems. Through benchmarking AI models—including convolutional neural networks (CNNs) and large language models (LLMs)—we evaluate AMM's impact on performance. We also explore challenges in AI model compilation, strategies for optimizing workloads, and future directions for RISC-V-based AI acceleration.

Introduction

Matrix multiplication is a fundamental mathematical operation that underlies many of today's most computationally intensive fields. It is essential in various domains, including deep neural network training in artificial intelligence, high-resolution image processing in computer vision, and complex equation solving in scientific simulations and modeling.

Recognizing the importance of accelerating matrix multiplication, RISC-V International (RVI) has established the Integrated Matrix Extension (IME) Task Group to develop standardized solutions. As an active participant, Andes Technology is contributing to these standardization efforts. However, to address the immediate market need for matrix multiplication acceleration, Andes has implemented the Andes Matrix Multiplication (AMM) instruction set. AMM extends the RISC-V Vector Extension (RVV), enabling customers to leverage enhanced matrix multiplication capabilities immediately. Andes has also shared this implementation in IME Task Group meetings, supporting the broader effort to define industry-wide matrix acceleration standards.

The AMM instruction set introduces two key features to optimize matrix operations. The first is a set of specialized Matrix Multiplication Instructions that support all combinations of signed and unsigned integer multiplication-accumulation. Currently focused on 8-bit operations, these instructions take two 8-bit values and generate a 32-bit accumulated result, with future plans to extend support for other data formats. Additionally, AMM allows flexible vector register grouping by supporting LMUL values of 1, 2, 4, and 8, ensuring adaptability to various computational workloads.

The second feature is a set of efficient Two-Dimensional Memory Access Instructions designed to streamline matrix data transfers between memory and vector registers. These instructions support both row-major and column-major data layouts, optimizing memory access patterns for improved performance. They also accommodate both 8-bit and 32-bit data transfers, providing versatility for different data structures.

AMM uses a dedicated user-mode control and status register (CSR) to enable precise control over partial matrix operations. This CSR includes ACTIVE_M, ACTIVE_N, and ACTIVE_K fields for zero-overhead boundary control, facilitating efficient handling of irregular matrix dimensions and partial calculations.

IREE (Intermediate Representation Execution Environment) [1] serves as a comprehensive compiler and runtime system built on MLIR technology. Its primary purpose is to transform ML models into a standardized intermediate representation that's versatile enough for large-scale datacenter deployments while remaining efficient for resource-constrained mobile and edge devices.

We evaluated AMM by incorporating it into IREE to compile and run ML workloads on RISC-V systems with AMM support. The integration process focused on enhancing IREE's compilation pipeline through several key developments:

1. Developing a new MLIR pass to tensorize matrix multiplication operation
2. Designing a custom MLIR dialect tailored for AMM to support all its instructions.
3. Implementing conversion passes to lower matrix multiplication and load/store operations from the vector dialect to AMM one

Our evaluation covered a broad spectrum of AI models, ranging from convolutional neural networks (CNNs) to large language models (LLMs), to measure performance improvements. This presentation explores the integration process, shares performance estimates, and discusses

strategies for optimizing AI workloads using AMM and RISC-V Vector Extensions. We also examine the challenges encountered during AI model compilation and consider future developments in RISC-V-based AI acceleration.

AMM on IREE

IREE Enhancements for AMM

To integrate AMM into IREE, two major enhancements were developed. The first enhancement focuses on selecting optimal tiling sizes (considering unrolling factors and register constraints), generating VLEN-agnostic code adaptable to different hardware configurations, and specifically addressing the distinct computational patterns of LLM prefill and decode stages.

The second enhancement involves the creation of a dedicated AMM dialect to manage AMM instructions. This new dialect serves multiple purposes: it contains comprehensive definitions for all AMM instructions and handles the conversion of vector dialect matrix operations into LLVM IR intrinsic functions. The dialect supports both scalable and fixed vector types, and includes an optimization pass specifically designed to eliminate unnecessary load/store operations, thereby enhancing overall performance.

Evaluation

For compiling PyTorch-based LLM models, we leverage the open-source project “shark-ai” to create the LLM model and add the KV-cache optimization. Shark-ai utilizes IREE Turbine, IREE's frontend for PyTorch, to compile the LLM into high-level Torch dialect.

We evaluated AMM's performance impact by compiling and benchmarking AI models, including MobileNet V1 (INT8) and TinyLlama (F32), on an Andes 64-bit RISC-V core supporting both RVV and AMM with a simulator. Results demonstrate significant speedups with AMM over standard RVV, achieving up to 6.4x for GEMM, 2.72x for MobileNet V1, and 2.02x for the TinyLlama prefill stage. We discuss the compilation workflow, optimization strategies employed, and the achieved hardware utilization, highlighting the effectiveness of AMM for accelerating diverse AI workloads on RISC-V platforms.