elliott binder

✉ ebinder@cmu.edu ☎ (814)-203-0173

phd candidate

Researcher in the modeling and design of high performance computation for CPUs, GPUs, and FPGAs, from single core to distributed processing.

education

Carnegie Mellon University
2017 to Current

PhD Electrical & Computer Engineering

MS Electrical & Computer Engineering 2019

QPA: 3.92

Johns Hopkins University
2012 to 2016

BS Computer Engineering 2016

GPA: 3.55

employment

AMD Research
Remote

Research Intern

Summer 2021

Investigating the benefits of potential future vector extensions for CPUs

AMD
Remote

MIOpen Intern

Summer 2020

- analyze the performance characteristics of convolutional neural networks on GPUs
- breakdown the implicit GEMM implementation into its fundamental operations
- identify how convolution parameters affect low-level code
- develop a performance model for convolution's computation and index calculation

IBM
Poughkeepsie, NY

Millicode Firmware Engineer

June 2016 to Aug. 2017

- develop embedded software for z Systems mainframes
- analyze and debug hardware traces from field and test environments
- design tools to expedite the development and debugging processes

PCTEST Engineering Laboratory, Inc.

Intern

Summer 2014

Carnegie Mellon University
Pittsburgh, PA

Research assistant

Sept. 2017 to Current

- analyze computational processes to determine their fundamental operations
- identify independent and dependent operations to guide implementation design
- model target architecture according to the hardware's capability
- evaluate algorithm and implementation design against state-of-the-art and theoretical peak

Los Alamos National Laboratory
Los Alamos, NM

Cyber Fire Intern

Summer 2018

- coursework in network archaeology, host forensics, malware analysis
- project in incident response involving analyzing network and host data
- present findings at an incident briefing to senior management

Johns Hopkins Applied Physics Laboratory

Intern

Summer 2015

projects

Accelerating ML-based Super-Resolution for Real-time Radar Signal Processing

Using Learned Iterative Shrinking Thresholding Algorithms (LISTA) as an example, we show the potential algorithm-software-hardware co-design opportunities for real-time machine learning inference. We explore accuracy and latency impacts of reduced precision, quantization, batching, and algorithmic changes.

Portable Implementations of Small Prime-Sized Discrete Trigonometric Transforms

Exploiting the symmetry found in the discrete Fourier matrix for prime-sized signals, we show how a well-design O(N^2) algorithm targeting modern CPU and GPU architectures can outperform state-of-the-art libraries and O(N log N) counterparts. Furthermore, we show how this implementation can easily be ported across CPU and GPU vendors, as well as used for real-to-complex, complex-to-real, cosine, and sine transforms with very little modification while maintaining competitive performance.

Analytical Model for Portable Performance of Matrix Multiplication-like Operations on GPUs

By analyzing the resource requirements of GPU matrix-multiply algorithms, we are able to derive software parameters from hardware features to produce high performance implementations. We show that this approach is portable across architectures and applications with similar data access patterns.

Distributed Multi-Dimensional Fast Fourier Transforms

Given the many potential ways to decompose a MDFFT, we leverage the knowledge of compute capability and communication costs of a distributed system to inform our implementation. To fully utilize this knowledge, we describe computation mapping from distributed nodes to individual GPU threads to reduce unnecessary data movement, minimize the effects of latency, and enable high-throughput computation.

Analytical Models for Deep Learning Recommendation Models on CPUs and GPUs

State-of-the-art implementations of DLRM utilize ML and linear algebra libraries to accelerate inference, but incur compute and data-movement overheads. We provide algorithms and analytical models for embedding, interaction, and MLP layers that are parameterized via hardware features.

A Portable GPU Framework for SNP Comparisons

We provide a framework for analyzing single nucleotide polymorphisms (SNPs) to detect the absence and/or presence of minor alleles, portable to a variety of GPU platforms. DOI: 10.1109/IPDPSW.2019.00041

Performance Design of Matrix Multiplications in Large Language Models

Considering the impact of the memory bound MMs present in multihead attention layers, we leverage our models of CPU and GPU architectures to design algorithms and kernels that differ from canonical high performance implementations for large input sizes.

Analysis and Acceleration of Fully Homomorphic Encryption

skills

LANGUAGES & TOOLS: C, C++, assembly, Python, CUDA, HIP, OpenCL, MPI, OpenMP, HLS, gem5, LLVM

RELEVANT COURSEWORK: How to Write Fast Code, Modern Computer Architecture and Design, Computer Architecture and Systems, Hardware Architectures for Machine Learning, Performance Modeling & Design of Computer Systems, Optimizing Compilers for Modern Architectures, Reconfigurable Logic: Technology, Architecture and Applications, Embedded Software Engineering, Secure Software Systems

APPLIED EXPERIENCES: Research in High Performance Computing, 2X teaching assistant for How to Write Fast Code, Optimization and modeling of dense linear algebra, convolutions, MLP layers, embeddings, interactions, Fourier transforms, genome analysis