BOYANA NORRIS

SR STAFF SOFTWARE ENGINEER (REMOTE)

Eugene, OR 97405 ♦ (541) 790-2027 ♦ brnorris03@gmail.com ♦ **WWW:** https://bold.pro/my/boyana-norris/757
♦ **WWW:** github.com/brnorris03 ♦ **WWW:** https://www.linkedin.com/in/boyananorris/

PROFESSIONAL SUMMARY

Organized, productive, and dependable candidate successful at managing multiple priorities with a positive attitude. Willingness to take on added responsibilities to meet team goals. Extensive experience in LLVM/MLIR-based compiler and runtime development for new AI architectures, high-performance computing (HPC) research on methodologies and tools for performance reasoning and automated optimization of scientific applications while ensuring continued or better usability of HPC tools and libraries and improving developer productivity. Developed multiple open-source software packages and coauthored over 100 peer-reviewed publications on topics including performance modeling, compiler-based performance optimization (autotuning), the embedding of domain-specific languages into legacy codes, source-transformation-based automatic differentiation, adaptive algorithms for HPC, and component-based software engineering.

SKILLS

Complex software design and implementation	LLVM/MLIR compiler development
Software/hardware co-design	CI/CD and build systems
Project management	Classroom instruction

WORK HISTORY

Sr Staff Software Engineer, 02/2024 - 08/2025

Tsavorite Scalable Intelligence, Inc – Milpitas, CA (Remote)

Design and implementation of an MLIR-based ML compiler and runtime system for Pytorch (or ONNX) models targeting custom parallel heterogeneous AI chiplets (CPU, vector, and matrix engines). Fully automated AOT and JIT compilation of unmodified Pytorch models (e.g., from Huggingface).

Sr Staff Software Engineer, 10/2023 - 01/2024

Rain Neuromorphics – San Francisco, CA

Design and implement an LLVM/MLIR-based optimizing compiler for ONNX ML models on Rain's parallel energy-efficient AI parallel tile-based architecture.

Sr Staff Software Engineer, 04/2022 - 10/2023

Luminous Computing Inc. – Santa Clara, CA

Design and implement parallelizing LLVM MLIR-based compiler for custom RISC-V-based AI architecture. Create SDKs to interface compiler with popular AI frameworks (PyTorch, Tensorflow) & runtimes. [Luminous ceased operations in Dec 2023]

Assistant/Associate Professor, 09/2013 - 09/2023

University Of Oregon – Eugene, OR

- Established the High-Performance Computing Laboratory, with research in performance analysis and optimization, compiler-based autotuning, and software engineering (funding: NSF, DOE, NIH, and industry).
- Created and taught undergraduate and graduate computer science courses, including CS1, CS2, Unix, C, C++, parallel computing, data science, and program analysis and transformation (advanced compilers).

Computer Scientist, 11/1999 - 08/2013

Argonne National Laboratory – Argonne, IL

Led the performance engineering group in the Mathematics and Computer Science Division and conducted research in performance analysis and optimization, automatic differentiation, and component-based software engineering.

EDUCATION

Ph.D.: Computer Science, 08/1995 - 01/2000

University of Illinois at Urbana-Champaign - Champaign, IL

Advisor: Prof. Michael T. Heath

Thesis title: An Environment for Interactive Parallel Numerical Computing

Bachelor of Science: Computer Science, 08/1993 - 05/1995

Wake Forest University - Winston-Salem, NC

No Degree: Computer Science, 08/1991 - 05/1993

Southwest State University - Marshall, MN

Languages

C++ — Excellent

C — Excellent

Python — Excellent

LLVM/MLIR — Excellent

Selected Software

ML Frameworks (Pytorch, TF, ONNX) — Very Good

Linux (any Unix) — Advanced

Parallel/Distributed (e.g., OpenMP, MPI) — Advanced

Build Systems (cmake, bazel, ...) — Advanced

Selected Software

- **Tsavorite ML** compiler is an MLIR-based compiler for AOT compilation of ML models (PyTorch or ONNX-based) for a heterogeneous scalable custom architecture that requires multi-target parallel code generation and optimization.
- **Luminous** ML compiler is a parallelizing MLIR-based compiler with support for both AOT and eager compilation of ML models implemented in TensorFlow or PyTorch (or any ML framework that can produce MLIR representations of model graphs). The MLIR pipeline includes both custom and open-source dialects and transformations and produces scalable parallel code targeting the Luminous accelerators.
- **Orio** (http://brnorris03.github.io/Orio/) is a lightweight, extensible open-source framework that supports the definition of embeddable domain languages and empirical performance tuning of C and Fortran applications. Orio employs a source code annotation approach that enables key computations to be expressed at a high level and embedded in existing code as comments, from which Orio generates many optimized versions, which are then evaluated empirically to select the best versions to use for production runs. Because the search space for nontrivial computations is prohibitively large, Orio incorporates several numerical optimization methods, including Nelder-Mead Simplex-based methods, genetic algorithms, and machine learning and has been cited hundreds of times and used in many different domains.
- **PBound, Mira, and Meliora.** Compiler-based static performance modeling tools: estimate the number of floating-point operations and memory accesses through source code analysis (C and C++), and provide upper bounds on the performance of an application; high-level user performance annotations for generating highly optimized code with the goal of increasing developer productivity, application

performance, and performance portability on high-end architectures. The latest tool, Meliora (https://github.com/HPCL/meliora), uses LLVM to extract a control flow graph-based program representation, augmented with instruction and memory use details at the basic block level, then uses a CNN to create an embedding that can be used to match codes to previously optimized implementations.

- **ADIC** is a source transformation automatic differentiation of ANSI C and C++ programs (http://www.mcs.anl.gov/adic). ADIC implements a technique for automatically transforming a computer code implementing an arbitrary mathematical function into another code that computes the function and its derivatives without incurring truncation error and often resulting in better performance than numerical approximation approaches, such as finite differences. ADIC has been downloaded thousands of times and has been used in numerical optimization, sensitivity analysis, climate modeling, computational fluid dynamics, and other application areas.