Keshav Vinayak Jha

keshavvinayakjha@gmail.com
linkedin.com/in/keshavvinayakjha/
+91-7678442691
github.com/keshavvinayak01
keshavvinayak01.github.io/

Education

- **Indian Institute of Science** — Bangalore
  Masters in Computer Science and Automation, GPA: 8.7/10.0 — *Aug. 2021 – Mar. 2024*

Research Interests

- Computer systems; compiler and runtime systems; GPU and accelerator systems; virtual memory management; memory hierarchy and performance analysis.

Publications

- **HyCache: Hybrid Caching for Accelerating DNN Input Preprocessing Pipelines**: **Keshav Vinayak Jha**, Shweta Pandey, Murali Annavaram, Arkaprava Basu. *USENIX ATC 2025*, pp. 433–448. Designed HyCache, an automatic hybrid memory/storage caching runtime for DNN input pipelines; improved raw pipeline throughput by 1.11×–10.1× across evaluated workloads.

Talks

- **Mitigating GPU Bank Conflicts with Swizzling**: Internal AMD technical talk on GPU LDS bank conflicts, padding, and XOR swizzling. Used IREE PR #23778 as a case study for compiler-controlled shared-memory layout, MI300X performance counters, and instruction-level trace analysis.

Experience

- **AMD (IREE Compiler Team)** — Remote
  Senior Software Development Engineer — *Aug'25 – present*
  - **Systems and Compiler Infrastructure**: Built infrastructure connecting PyTorch graph programs to MLIR/IREE lowering and runtime-facing metadata, focusing on correct IR semantics, backend legality, and robust end-to-end compilation.
  - **GPU Performance and Code Generation**: Implemented target-aware vector rewrites, math-library lowering, and LLVM GlobalISel backend improvements for ROCm/AMDGPU, with emphasis on correctness and code-generation quality.
  - **Program Analysis and IR Optimization**: Designed dialect extensions, canonicalizations, folders, and peephole optimizations across multiple IR layers to simplify programs before lowering and preserve compiler invariants.
  - **Compiler-Driven Tuning**: Extended tuning infrastructure for GPU workloads by encoding kernel-generation constraints such as accumulator types, denormal handling, and vector distribution strategies.
- **Cerebras Systems** — Bangalore, India
  Member of Technical Staff — *March'24 – July'25*
  - **Compiler Infrastructure**: Developed MLIR/LLVM and torch-mlir based compiler infrastructure for a non-traditional AI accelerator, working across Python frontends, PDL-based rewrites, tiered dialects, and low-level chip IR.
  - **Abstraction Lowering**: Implemented compiler rewrites, canonicalizations, and legality-preserving lowering logic across multiple IR layers to improve model coverage, maintain IR invariants, and support backend code generation.
  - **Systems Engineering**: Worked in a production compiler codebase where correctness, debuggability, regression testing, and long-term maintainability were required for large model workloads.
  - **Open Source**: Upstreamed torch-mlir LTC backend patches for shape inference and operator support under `keshavj-cerebras`.
- **Indian Institute of Science** — Bangalore, India
  Graduate Researcher — *Nov'22 – Mar'24*
  - **GPU Resource Partitioning**: Studied mechanisms for co-running DNN workloads on a single GPU using NVIDIA MIG and SM partitioning, analyzing how resource partition sizes and workload characteristics affect isolation, utilization, and end-to-end performance.

- **Multi-DNN Memory Management**: Explored DNN checkpointing strategies for concurrent single-GPU execution, trading recomputation for lower activation-memory footprint and improved ability to co-locate multiple models under memory pressure.
- **Systems Measurement**: Built performance-analysis workflows for GPU workloads, using profiling and microbenchmarking to reason about compute utilization, memory pressure, interference, and scheduling behavior.

Selected Systems Projects

- **Multi-Level Intermediate Representation (MLIR)**:
  - **Loop optimization:** Implemented an MLIR Affine loop-interchange pass driven by an analytical locality cost model covering spatial, temporal self, and temporal group reuse. The pass handles perfectly nested loop nests and conservatively bails out on conditionals to preserve correctness.
  - **Accurate reductions:** Designed MLIR infrastructure and compiler passes for atomic reduction operations on CPU targets. The passes detect sequential read-modify-write patterns and replace them with atomic operations in-place; source available here.
- **Performance analysis of AMD and Intel processors**: Analyzed SPEC CPU2017 workloads using `perf` and hardware counters to compare microarchitectural behavior across processors. Studied normalized cycle count, IPC, branch prediction, L2 DTLB hit rate, L2 cache behavior, and wasted work; also designed a branch-misprediction microbenchmark that achieved a 33% miss rate. Report available *here*.
- **Ballooning subsystem in memory management for Linux**: Modified Linux kernel v5.11.5 memory management to add an application-facing memory-ballooning API. Added kernel-level system calls, monitored memory pressure, and signaled registered applications to release memory under pressure. Patch and commits available *here*.
- **Matrix Multiplication**: Optimized GEMM on CPU/GPU using hardware-counter-guided analysis. Achieved a 1017× speedup over a single-threaded CPU baseline using a GTX 1650 kernel with scratchpad tiling; CPU optimization used loop blocking, packing, tiling, and AVX-512 vectorization. Report available *here*.

Coursework

- **Computer Architecture:** 10/10, highest in batch; taught by Prof. Arkaprava Basu.
- **Advanced Compilers:** 9/10, highest in batch; taught by Prof. Uday Bondhugula.
- **Operating Systems:** 8/10, highest in batch; taught by Prof. Arkaprava Basu.
- **Linear Algebra and Probability:** 8/10.

Achievements

- **Academic**: GATE CS'21 AIR 390, Speaker at USENIX ATC'25

Programming Skills

- **Languages**: Python, C/C++, SQL, HTML/CSS, JS
- **Systems and Compiler Tools**: LLVM, MLIR, IREE, torch-mlir, PyTorch, ROCm, perf, Git, C++ STL