Vedant Kumar

vk@vedantk.com

Strengths

Systems programming and code review; written and verbal communication; working across teams. I'm proficient in: C/C++, Python, and shell scripting; solving performance/correctness issues in large, complex codebases; LLVM development. I'm able to learn quickly and enjoy being a mentor.

Experience

- Apple, 2021-present, Senior Engineer, Kernel and Runtime
  - '20-'22: Worked with Apple's silicon engineering, kernel, and devtools teams to develop hardware trace technologies. I built and deployed CPU trace-based performance analysis and debug tools, consulted on hardware projects, and helped teams at Apple gain deeper insights into critical workloads.
- Apple, 2019-2021, Senior Engineer, Language and Runtime
  - '19-'20: Worked within a small, select team to address critical issues in Apple's developer tools on the then ultra-secret macOS Apple Silicon project. We delivered on time and on a tight schedule, facilitating a smooth transition within the company when the project was made public at WWDC '20.
- Apple, 2015-2019, Engineer, Language and Runtime (LLVM/Clang)
  - '18-'19: Worked with the iOS performance team to reduce the memory footprints of key applications and system processes. Implemented a compiler optimization to outline and reorder cold failure paths. Successfully pushed for adoption of link-time optimization (LTO) in tens of projects using an evidence-based approach, demonstrating memory savings and performance improvements to project owners. Implemented a new dyld cache ordering algorithm, reclaiming 5-20MB of memory on embedded devices. Worked 1:1 with tens of engineers from different teams during office hours, to help investigate and fix excessive memory usage issues.
  - '17-'18: Improved optimized code debugging. I fixed bugs causing incomplete backtraces, incorrect line tables, or missing variable locations. I worked on clang, swift, LLVM's mid-level passes, its AArch64 and x86_64 backends, and the debugger. I created tooling to isolate debug info quality regressions and mentored a GSoC student working with it.
  - '16-'17: Added and fixed diagnostics in Undefined Behavior Sanitizer (UBSan), a widely-deployed run-time bug detection tool. I integrated UBSan into Xcode's build system and source editor, bringing UBSan to a large new audience. I presented this feature in a WWDC '17 talk, contributed documentation, and helped teams at Apple eliminate UB in their projects.
  - '15-21: Maintained the source-based code coverage implementations in Swift and Clang. I improved the precision and quality of coverage reporting, documented the feature, set up continuous integration testing for `-fcoverage-mapping`, substantially reduced its overhead, and reviewed and merged coverage-related patches from new contributors. Members of the LLVM community rely on our bot's reports.

  - '15-'16: Helped stabilize several versions of Clang. These compilers built Apple's entire software stack (firmware, kernel, frameworks etc.) and were shipped to millions of developers.

- Apple, 2013-2014, Summer Intern (2x), Filesystems team
  - Wrote fuzzers to uncover and fix race conditions in XNU.
  - Top-five finalist in the Apple-wide software engineering intern competition (2013).
- UC Berkeley, 2015, Teaching Assistant, Programming Languages and Compilers (class website)
- UC Berkeley, 2014, Teaching Assistant, Advanced Operating Systems (class website)
  - Our team wrote an EDF scheduler, a small filesystem, and a device driver for Linux.
- HP Fortify, 2012, Summer Intern, Static Analysis team
  - Designed, implemented, tested, and shipped an interprocedural constant propagator.
- Personal projects (C++, Python), 2009-2014. All source code is available here.
  - Data structures: skiplist, heaps, trie, B+ tree, kd-tree, open-addressed hash table
  - *quotient-filter*: compact approximate membership filter, supports merging and deletions
  - *53otron*: primitive Lisp $\rightarrow$ LLVM compiler, used to vectorize equations to draw 3d shapes
  - *auto-diagonalize*: optimization pass to convert linearizable loops into $\Theta(\log n)$ processes [1]
  - Graphics: a fast ray-tracer (*radiate*), a model controlled with inverse kinematics (*spike*)
  - Networks: an `epoll`-based server (serves `vedantk.com`), a multi-threaded server

Education and Academic Honors

- UC Berkeley (2011-2015): B.S in Electrical and Computer Engineering.

  At Berkeley, I served as a TA twice and participated in lightweight crew for a semester. I volunteered to teach several practical programming workshops, including a workshop on Unix tools and one on LLVM development. I took Advanced Operating Systems, Intro and Graduate CS Theory, Computer Architecture and Engineering, Discrete Math and Probability, Intro to Computer Graphics, and Intro to Computational Biology.

- International Science and Engineering Fair Finalist (ISEF 2009 and 2010). Awarded a trip to CERN and an asteroid.

[1] I demoed the optimization pass at the 2014 LLVM developer meeting: slides, longer writeup.