Austin Hicks

Location: Seattle
Phone: 561-308-1476
E-mail: ahicks@ahicks.io
Home page: https://ahicks.io
Twitter: ajhicks1992
GitHub: ahicks92

Programming languages
Rust, C/C++, Python, JS, bit of golang, OpenResty Lua, and Julia

Skills
Reliable microservice design, Rust async, sql (sqlite and postgres), low-level performance optimization, bug-free C/C++, Software design for on-prem environments, Kubernetes, Docker, GCP, BigQuery, Django, Postgres, RabbitMQ

Education
Graduated Florida Atlantic University Summa Cum Laude with a BS of Computer Science, class of 2015. One of two students recognized by the president of Florida Atlantic University for outstanding achievement.

References available upon request.

Highlights
- Can reliably produce working and performant C/C++ code for release on a weekly cadence, using tools such as address sanitizer.
- Have worked on and helped design highly parallel systems in C/C++, including the use of epoch-based reclamation and ConcurrencyKit.
- Have regularly worked on and designed code which must run in on-prem environments with no direct access.
- Designed and implemented multiple production-grade microservices, some handling terabytes of data and 20k+ requests per second, which can go months at a time without needing human intervention.
- Designed and implemented a bespoke service discovery and clustering solution.
- Designed and implemented a custom microservice broker/monitoring solution capable of handling 10000 requests per second.
- Made significant contributions to the Rust compiler.
- Designed, implemented, and helped to maintain multiple billing systems.
- Have significant experience writing mathematical code in C++ and Rust, including custom parallelization algorithms and SIMD optimizations.

Work Experience

Sauce Labs/Backtrace I/O (Staff Software Engineer, Nov 2019n – present)
- Part of Sauce Labs' acquisition of backtrace I/O in July 2021. Backtrace is a product for receiving, analyzing, and querying crash data focused on native crashes (AKA Sentry for C++).
- Designed, implemented, and released a variety of features for an error management data warehouse used by many AAA game publishers and large mobile app developers.

- Owned and lead projects from initial idea to successful completion, including requirements discovery, tech specs, and implementation.
- Worked on a team which releases reliable C code on a weekly cadence.
- Owned a Rust codebase and developed multiple Rust microservices, some of which go months without devops intervention.
- Worked on improving and parallelizing our custom columnar database.
- Notable projects:
  - One of two engineers who proved and spearheaded the adoption of Rust at our organization, leading to adoption across the stack. As a result of this work, our core C codebase is slowly being converted to Rust.
  - Implemented service clustering support. Service owners need only flip a flag, write a couple small service-specific methods, and enable it in the config file.
- Implemented support for user sessions, a service which receives a request every time an application using our SDK is launched to provide analytics, capable of handling around 100 million requests per day on only 5 nodes.
- Reimplemented our symbol processing pipeline, the service which converts user binaries into an internal proprietary format, centralizing a large amount of compute and improving reliability for on-call engineers.
- Migrated our initial Rust services from sync (Rocket) to async (Tokio and Warp) Rust.
- Implemented our feature flag system which uses self-modifying assembly to provide flags for C codebases that can be used even in tight loops.
- Designed and Implemented support for sending alerts to clients when they receive too many errors, backed by our freeform query engine.
- Implemented support for server-side automation such as notifying users when an error is seen in a new version of their application, automatically assigning errors to specific teams, and adding categorization metadata.
- Added support for fine-grained server-side error rejection and sampling, which customers can use to both reduce data storage requirements and to shed load.
- Assisted in the design and implementation of a migration from local storage to S3 across the stack to reduce costs.

Cielo24 (Research and Development, 2017-2018)

- Tested an implementation of distributed locks, proved that it was nonfunctional, and replaced it with a Redis-backed solution.
- Implemented a bespoke microservice proxy, including the ability to view all requests in real-time and link them back to the user that triggered them.
- Implemented a massive autoscaling media transcoding cluster powered by the kubernetes jobs API which ran hundreds of thousands of jobs per day.
- Wrote a custom extract-transform-load data processing pipeline to move large amounts of data from legacy systems to Google BigQuery for machine-learning-related exploratory analysis.
- Administered PostgreSQL:
  - Planned and executed a cross-cloud PostgreSQL migration of a large database.
  - Managed a deployment of PostgreSQL on top of Kubernetes using Stolon. Configured proper health checks and read replicas.
  - Performed query optimization using the slow log, explain, and hypothetical indexes.
  - Rolled out and administered PgBouncer, a connection pooling solution for PostgreSQL.
- Administered RabbitMQ, including setting up multi-datacenter HA and performing a cross-cloud migration.

- Implemented a custom, low-latency priority queue on top of Redis responsible for assigning tasks to human workers, including task priority updates, task cancellation, task preview, and task timeouts.

FutureTech Industries (Software Engineer, 2018-2019/part time 2016-2017)

- Interfaced directly with clients to determine requirements, refine stories and tickets, and address client concerns.
- Designed and implemented 2 billing solutions.
- Wrote a production-quality Helm chart for PgBouncer and coauthored articles explaining its proper configuration.

Rust compiler Development (2016 - 2017)

- Implemented an optimization with proven real-world impact that reduces the size of Rust data types by reordering fields to eliminate padding.
- Removed significant duplication in the compiler's internal representation of data types, a precursor to further work done by others, such as `repr(transparent)` and additional niche-filling optimizations for enums.
- Introduced the concept of optimization fuel, which can be used to binary search the compiler's decisions to find bugs.

Libaudioverse (open-source side project, C++, author and primary contributor, 2014-2018)

- An open-source DSP library written in C++ for audio synthesis, not dissimilar to WebAudio. Design independently converged with the WebAudio spec later in the library's lifecycle.
- Built releases with Appveyor and Travis CI based off git tags.
- Implemented a node execution engine capable of parallelizing audio synthesis work.
- Implemented the basic building blocks necessary for audio synthesis, including optimized waveform generators, convolution nodes, HRTF, reverb, and white/brown/pink noise implementations.
- Wrote optimized mathematical kernels for convolution and other DSP-related tasks.
- Wrote audio I/O code with support for multiple backends.

DictationBridge (Open-source C/C++, developer, 2017)

- Implemented the initial version of a solution to get JAWS for Windows (a screen reader) talking to Dragon Naturally Speaking (a speech recognition solution), enabling visually impaired individuals with additional physical disabilities to more easily utilize computers.

Simplemail (Florida Atlantic University Softwaer Engineering Project, Team Leader/Head Developer, 2015)

- Led 4 other students in the design and development of a prototype-quality mail inbox provider as part of CEN 4010 Principles of Software Engineering.
- Set up and administered GitHub repositories, domain names, servers, and other infrastructure for my team.