Rustin Liu

✉ rustin.liu@gmail.com · ☎ +(86) 189-0839-4732 · ⌨ Rustin170506 · ⚲ rustin.me

🎓 Education

Software Engineering Bachelor's Degree   Sep 2015 – June 2019

- Chongqing University of Technology

⚙ Skills

- **Programming Languages**: Experienced in Go/Rust/JavaScript, **multilingual** (can adapt to any language)
- **Development Tools**: familiar with Linux-based programming, have experience with tools like Git, GitHub, Docker, Jenkins, Github Actions and Jira, etc.
- **Languages**: **English** (Can read, write and speak fluently), **Chinese** (Native speaker)

👥 Work Experience

PingCAP Inc. - Database - Database Kernel R&D(Golang/Rust)   August 2023 – Present

Transferred from the data platform department to the kernel computing department and started to focus on the development of the optimizer of the distributed database TiDB.

Responsibilities:

- Designed, implemented and maintained the optimizer of the distributed database TiDB.

Accomplishments:

- Optimized statistics collection by implementing an in-memory priority queue, eliminating repeated queue rebuilding overhead, reducing CPU usage by one core and memory consumption by 99%.
- Designed and implemented the predicate columns feature to optimize statistics collection by only analyzing columns used in predicates, reducing auto analyze time by 50% in OLTP workloads and nearly 99% in partitioned wide table scenarios.
- Designed and implemented the priority queue for speed up the auto analyze process, significantly reducing the starvations of the auto analyze process.
- Added support for the LOCK STATS TABLE feature to allow users to lock statistics updates and generate stable execution plans.

PingCAP Inc. - Database - Database Tools R&D(Golang/Rust)   July 2021 – July 2023

Transferred from the community department to the data platform department and started to focus on the development of data synchronization tools TiCDC.

Responsibilities:

- Designed, implemented, and maintained core modules, focusing on Sink and Sorter, and managed daily project maintenance including issue tracking, code reviews, and on-call customer support.

Accomplishments:

- Changed push mode to pull mode to optimize memory management and data synchronization process, greatly improving OOM issues and synchronization lags.
- Modified the Sink module from synchronous to asynchronous mode to optimize code performance and improve throughput by 30% in incremental scenarios.
- Added multi-topic and OAuth authorization support for Kafka Sink, greatly reducing the operation and maintenance costs of multi-topic tasks.
- Added the ability to optimize the Lossy DDL from full table scan to not synchronizing any data, greatly reducing the impact of DDL on the system.
- Refactored all of TiCDC's CLI code to standardize the way commands are added and written, which greatly improved maintainability.

PingCAP Inc. - Database - Full-Stack Engineer(Golang/JS)   Aug 2020 – July 2021

Joined the community department to develop TiChi to support community collaboration based on the Kubernetes Community Prow project.

Responsibilities:

- Designed and implemented a project with collaborative robots on GKE using Prow, optimizing community collaboration processes for open source contributors.

Accomplishments:

- Completed TiChi, which has become the company's CI/CD infrastructure, on time.
- Fixed bugs and added features for Prow, submitted 39 PRs.

Morningstar, Inc. - Financial Services - Full-Stack Engineer(Java/JS)   June 2019 – July 2020

First job after graduation, responsible for writing and maintaining data presentation components, left after the company decided to start closing its China R&D center.

Responsibilities:

- Developed and optimized front and back-end components in Java and Vue, enhancing engineering efficiency and front-end infrastructure.

Accomplishments:

- Independently developed components that are already online, with good stability and performance.
- Implemented a front-end tracking system that uses HeapIO to track user information, with an extensible interface to seamlessly switch to Google Analytics.
- Wrote automated Jenkins scripts for component and document releases, optimizing the release process from a manual 4-hour process to a fully automated 30-minute process.

🐙 OPEN SOURCE PROJECTS

Cargo(Rust) - Maintainer   260+ commits

The Rust package manager. Cargo downloads your Rust package's dependencies, compiles your packages, makes distributable packages.

- Implemented the 'cargo-info' subcommand to display detailed information about a Rust package.
- Improved various error prompts for Cargo commands and the Rust compiler to make errors to effective and clear.
- Added the –crate-type function and added related documentation for Cargo.
- Added support for automatic inheritance of the [workspace.package] fields for the cargo new/init commands.
- Added new syntax for Cargo build scripts to provide a foundation for future extensions to build scripts.
- Added 'target.'cfg(..)'.linker' configuration option to allow users to specify the linker via 'cfg(..)' conditions.
- Added OR_PATTERNS_BACK_COMPAT lint to help users migrate to the new orpatterns syntax.
- Made authors field optional in Cargo and rustdoc.

crates.io(Rust/JS) - Maintainer   38+ commits

The Rust package registry.

- Helped implement RFC 3052 to allow crates.io to accept packages without author information.
- Added support for renaming packages with underscore prefixes to be published on crates.io.
- Aligned crates.io and Cargo on the naming rules for features and crates.
- Implemented token expiration notifications and one-click token copying to improve user experience.

console(Rust) - Maintainer   283+ commits

A debugger for async rust!

- Implemented grpc-web support, enabling usage in web browsers.
- Added WatchState API to accurately manage the state of different clients.
- Correctly handled the display of paths on Windows.   ·

tokio-console-web(TypeScript/Rust) - Author   283+ commits

tokio-console-web is a web-based console for debuging and monitoring the tokio runtime.

- Displays all tasks in the tokio runtime and their status.
- Displays the details of each task, including a visual histogram of the polling(busy) times of the task.
- Displays the resources, such as synchronization primitives, I/O resources, etc., used by each task.

Rustup(Rust) - Previous Maintainer   110+ commits

Ant Design(TypeScript) - Previous Maintainer   20+ commits