Rust: A Systems Programming Language with Modern Comforts

Tyler Nienhouse – CSCV 372 – 12/04/2016

Overview

Rust is a systems programming language designed for safety, concurrency, and speed.[1] It's development is driven by the idea that not everything must be a trade-off.[2, 13:00] Rust's primary unique feature is its resource ownership and borrowing scheme.

The following research centers around three major topics – a high level of discussion of what Rust does to accomplish its goals, various companies' experience using Rust in their production code environments, and how the resource ownership system affects other aspects of the language.

What Makes Rust Special

Rust's major competitors are C and C++, with its syntax being based on C/C++ as well as ML/Haskell.[3, 1:25] What really sets Rust apart from other languages, though is its semantics and community – rather than the syntax.

Memory Management[3]

Unlike many other modern compiled languages, Rust uses deterministic destruction of resources instead of Garbage Collection (which would require an inefficient runtime). Rust provides deterministic destruction with its ownership system:

- **Ownership**: Only one party/binding owns a given resource (stack data, heap data, files, etc.) at any given point of a program. Ownership can be transferred between parties and the resource is destroyed when the most recent owner goes out of scope.
- **Borrowing**: The owner of a resource can delegate two types of borrows:
  - **Shared**: Multiple parties may only access the resource that has been shared to them, no mutation is permitted during shared borrows.
  - **Mutable**: A single party may access and mutate the data, but no other parties are permitted to access or mutate the data simultaneously.

Modern Offerings

Rust also provides a modern set of tools both within the language and as additional officially maintained applications to set it apart.

- **Functional Programming**: Rust provides many features characteristic to functional programming languages[2, 21:40] like lambdas/closures and higher-order functions.
- **Cargo**: A build and dependency management tool that replaces many uses of Makefiles for building as well as provides access to the official Crates.io repository of prebuilt libraries for use in projects.[1]
- **Rustup.rs**: The Rust toolchain installer that provides easy access to downloading, installing, and updating the Rust binaries and source.[1]

MaidSafe

Re-wrote their C++ distributed storage app entirely in Rust.[4, 27:15]

• Quick to develop with • 10x reduction in codebase size	• Loss of support from C++ code vendors

Dropbox

Wrote their storage and compression service in Rust.[4, 36:40]

• Safety and correctness required for storing users' files	• The compiler becomes slow on sizable codebase

mozilla

Shipping Rust code in recent Firefox releases.[5]

• Zero errors in over 1 Billion executions	• Work required to integrate with C++ build system

coursera

Using Rust to manage automated assignment grading.[6]

• Safely execute untrusted code • Easy to maintain	• Only a niche component in a large stack

SKYLIGHT

Used to write a Ruby application performance monitoring tool.[4, 46:10][7]

• No segfaults or resource leak • No GC pauses affecting stats	• Steep initial learning curve for developers

Discussion

Reliability

Reliability is Rust's key focus, and many of the production users agreed that this was indeed achieved. Thanks to its strong typing system, Rust prevents common memory errors like dangling pointers, aliasing, and invalidated iterators.[3] This class of bugs is one of the most common critical security issues.[4, 52:00]

Costs

Readability / Writability: Rust's resource ownership paradigm can leads to the language having a fairly steep learning curve.

Time: Using the compiler to enforce the ownership system and resolve abstractions adds a decent compiling time cost to using the language.

These costs are often quickly recovered thanks to the lack of time spent on debugging hard-to-find memory errors.

Concurrency

Rust's memory management strongly resembles major techniques used for concurrent programming. For example the mutable borrow resembles the concept of using a Mutex, and the shared borrows are much like a common system of readers and writers in a database. Having these concepts in the language makes concurrency easy.

Conclusion / Future Discussion

While it is unlikely that larger companies will abandon their C and C++ code like MaidSafe has[4, 27:15] or Mozilla plans to[5], Rust has definitely found a strong starting point as a niche language that provides systems-level performance to groups that have stringent safety and correctness requirements.

Rust is still quite a young language, with its first stable release less than two years ago, and while it has seen adoption by several major companies it still has a ways to go before reaches the mainstream. Rust's use in production will likely continue to be a hot topic in the years to come.

References

1. "The Rust Programming Language." Official Rust Documentation. 22 Nov 2016.
2. Turon, Aaron, Niko Matsakis. Opening Keynote. RustConf 2016. 10 Sep 2016.
3. Turon, Aaron. "The Rust Programming Language." Colloquium on Computer Systems Seminar Series. Stanford University. 11 Mar 2015.
4. Klabnik, Steve. "Rust in Production." Philly ETE. 12 Apr 2016.
5. Herman, Dave. "Shipping Rust in Firefox." Mozilla Hacks. 12 Jul 2016.
6. Saeta, Brennan. "Rust & Docker in production @ Coursera." Coursera. 7 Jul 2016.
7. Katz, Yehuda. "Rust Means Never Having to Close a Socket." Skylight.io. 7 Oct 2014.