Exploring Rust in High-Performance Computing for Mitigating Errors and Improving Security

Jake Tronge, Howard Pritchard

July 27, 2022

Abstract

Most existing HPC applications and libraries are written in C, C++ or Fortran. Research into programming errors and security vulnerabilities have revealed that many of the errors that appear in programs written in these languages are related to memory errors, such as buffer-overflows, double-frees and data corruption. These can be traced back to the fact that these languages perform little to no checks which ensure that memory is being properly used within code. Within the past decade there has been a lot of research into using "memory-safe" languages; these perform checks either at compile-time or at runtime that are designed to ensure that memory is being used correctly. This eliminates many, if not all, of these types of memory-related issues that plague most low-level and performance-driven software. In HPC, since testing and debugging is made many times more complicated by the need for parallelism and the use of specialized hardware, memory-safe languages may offer some serious improvements to development and security issues. Rust is one promising memory-safe language that was originally developed by the Mozilla Foundation for improving memory-safety in the Firefox web browser, but has since spread to many other projects and organizations. Our work here is looking into how Rust can be used in HPC, comparing it with existing codes through benchmarking, as well as finding places where Rust might be able to offer improvements one component at a time. Initially we've taken a look at improving existing MPI-bindings to Rust and noting where Rust diverges from existing languages. In future work we plan to research other ways where Rust might improve HPC software and development.

LA-UR-22-27485