Improving MPI with Rust

Abstract

Jacob Tronge

LA-UR-23-28913

MPI continues to play an important role in HPC applications. At the same time, existing implementations fail to provide guarantees of safety and correctness, instead leaving many important error conditions to be checked by the user. Many of these errors, such as mismatched types or mismatched collective arguments, can lead to memory corruption, segmentation faults, and undefined behavior, which typically comes from a lack of memory safety guarantees. From a different perspective, the MPI implementations themselves often encounter similar problems but from within the library itself. Implementations are increasingly required to adapt to new hardware and programming environments, requiring extensive development and testing that can leave room for memory safety related errors. One way to solve these problems is to work with newer languages that are designed to guarantee memory safety: the Rust programming language is one such language that attempts to guarantee memory safety while maintaining performance close to that of C. In this presentation I'll give a brief overview of two different prototypes written in Rust that attempt to solve these problems and then show results that indicate that performance is close enough to the original versions to merit further consideration of Rust for HPC applications.

Los Alamos
NATIONAL LABORATORY