Agile Acceleration of LLVM Flang Support for Fortran 2018 Parallel Programming

Katherine Rasmussen1, Damian Rouson1, Najé George3, Dan Bonachea1, Hussain Kadhem1, Brian Friesen1
1Lawrence Berkeley National Laboratory, 2San Diego State University

Introduction

Problem

 LLVM/Flang Fortran compiler is currently Fortran 95 compliant, and the fortran can parse Fortran 2018. However, Flang does not have a comprehensive 2018 test suite and does not fully implement the static semantics of the 2018 standard.

Solution

 Agile software encourages early delivery of working software subject to continual improvement. We are investigating whether agile techniques centered around pair programming and test-driven development (TDD) can help Flang to rapidly progress to Fortran 2018 compliance. Because of the paramount importance of parallelism in high-performance computing, we are focusing on Fortran’s parallelism’s, called “Caffeine.”

Objective

• Exhaustively delineate all of the parallel programming features in Fortran 2018
• Develop semantics tests for LLVM Flang to cover statically checkable program errors that the Fortran standard obliges the compiler to detect
• Expand front-end support, including additional error checking, when tests identify missing capabilities

Approach

• Employ agile software development practices
• Test a comprehensive range of standard-conforming and non-conforming Fortran 2018 syntax
• Test-driven development: any contributed tests that fail provide a specification for new features to add to Flang

Agile Development

• Pair programming sessions
• Valuable team member interactions
• Get feedback early and frequently
• Leverage existing git and Github tools
• Leverage existing agile practices of the LLVM developer community
• Use LLVM’s continuous integration CI test infrastructure to quickly fix CI failures
• Code reviews on Phabricator for feedback, edits, and approvals

GitHub Project Board

Test-Driven Development Example

• Because Flang cannot yet produce executable files from Fortran 2018 source code, we are developing runtime tests in a separate repository called Caffeine
• Caffeine is a runtime library that supports parallel Fortran 2018 features.
• Caffeine runs atop the GASNet-EX exascale networking middleware.

Outcomes

• The Berkeley Lab fork of the LLVM-Project GitHub repository includes a project board capturing an exhaustive list of 41 parallel features to test.
• We have pushed static semantics tests for 22 such features upstream to LLVM-Project; intrinsic functions supporting parallelism, collective subroutines, atomic subroutines, synchronization statements, and more.
• We have contributed additional static semantic analysis and error checking for 11 missing parallel features exposed by our tests. More contributions are under development or in code review.
• We contributed cross-checks for 2 non-parallel features.
• We have developed 44 runtime tests that serve evidence by developing Caffeine.

Figure 1: Exhaustive list of Fortran 2018 parallel programming features to test https://go.lbl.gov/flang-testing

Figure 2: Diagram outlining the components of the static semantic tests for intrinsic functions and intrinsic subroutines

Figure 3: Main Caffeine repository for the LLVM Fortran Compiler

Figure 4: Static semantics test except for the __ws__ intrinsic subroutine, this test expectedly fails

Figure 5: Updated static semantics test excerpt for the __ws__ subroutine that passes after interface is added

Figure 6: Interface to compiler for __ws__ allows the test to pass when combined with a static semantic check for combined objects (not shown).

Figure 7: Signatures for the intrinsic collective subroutine, __co_sum__, as defined by the Fortran 2018 standard. It is the only required argument and the test of the arguments is optional.