Accelerating Fortran `DO CONCURRENT` in GCC

Google Summer of Code 2022 Proposal

Wileam Yonatan Phan, Contributor
Tobias Burnus (Mentor Graphics / GCC), Mentor

Abstract

`DO CONCURRENT` is a parallel execution control construct introduced in the Fortran 2008 standard. This project entails (1) adding GCC support in for Fortran 2018 locality specifiers; (2) adding GCC support for the Fortran 202X REDUCTION clause; and (3) implement parallelization on both multicore CPU and GPU devices controlled by the new `` `-fdo-concurrent` `` compiler flag.

Background

Fortran is one of the first high-level programming languages in existence. Created by John Backus at IBM in 1957 as FORmula TRANslation (FORTRAN), the language has a foothold in high performance computing (HPC) and scientific computing, especially in the physical sciences. Key strengths of Fortran include the natural syntax for expressing array computations, as well as being the only ISO-standardized language with built-in native parallelism.

In the past decade, accelerated computing – offloading computation to accelerator device(s) – has become more commonplace. Most often, the accelerator device is a graphics processing unit (GPU). There are two major hardware vendors (NVIDIA and AMD) that currently support this programming paradigm, with a third (Intel) coming soon. In addition, portable directive-based programming models also exist (notably, OpenMP and OpenACC) that can be used to offload computation to the GPU.

`DO CONCURRENT` is an execution control construct introduced in the Fortran 2008 standard. This construct has been enhanced significantly in the Fortran 2018 standard with the addition of three locality specifiers (`LOCAL`, `LOCAL_INIT`, `SHARED`) and the `DEFAULT` clause for defining the default locality of variables used in the construct, and will be enhanced further in the upcoming Fortran 202X standard with the `REDUCTION` clause that allows reduction operations. Stulajter et al. [arXiv:2110.10151] explores the current available Fortran compiler support for `DO CONCURRENT`. Three compilers currently support it: Intel `ifort`, GCC `gfortran`, and NVIDIA `nvfortran`. In addition, the latter is able to offload `DO CONCURRENT` loops to NVIDIA GPUs.

Currently, GCC supports GPU offloading through OpenMP `target` and OpenACC to both NVIDIA and AMD GPUs. This project aims to add a third avenue to accelerated computing: Fortran `DO CONCURRENT` construct with GPU offloading.

Scope

For the duration of Google Summer of Code 2022, we will work on the following tasks:

1. Add support for the Fortran 2018 locality specifiers and the `DEFAULT` clause in the GCC Fortran parser.
2. Add support for the upcoming Fortran 202X `REDUCTION` clause in the GCC Fortran parser.
3. Implement both CPU-based and GPU-based parallelization of DO CONCURRENT as controlled by the new `` `-fdo-concurrent=…` `` compiler flag. The possible options to this compiler flag are:
    - `serial` - no parallelization.
    - `parallel` - pthreads-based CPU parallelization similar to/based on the existing `` `-ftree-parallelize-loops=n` `` compiler flag.
    - `openmp` - CPU parallelization using the OpenMP programming model.
    - `openmp-target` - GPU offloading using the OpenMP programming model.
    - `openacc` - GPU offloading using the OpenACC programming model.