Solving Sparse Linear Systems of Equations using Fortran Coarrays

Ambra ABDULLAHI HASSAN [a], Valeria CARDELLINI [a] and Salvatore FILIPPONE [b]

[a] *University of Rome Tor Vergata*
[b] *Cranfield University*

Abstract. Coarrays have been part of the Fortran standard since Fortran 2008 and provide a syntactic extension of Fortran to support parallel programming, often called Coarray Fortran (CAF). Although MPI is the de facto standard for parallel programs running on distributed memory systems and little scientific software is written in CAF, many scientific applications could benefit from the use of CAF. We present the migration from MPI to CAF of the libraries PSBLAS and MLD2P4 for the solution of large systems of equations using iterative methods and preconditioners. In this paper we describe some investigations for implementing the necessary communication steps in PSBLAS and MLD2P4 and provide performance results obtained on linear systems arising from discretization of 2D and 3D PDEs.

Keywords. CAF, MPI, numerical linear algebra, sparse linear systems

1. Introduction

Fortran has been the dominant language in High Performance Computing and still is a very important player in the field. In past couple of decades, the language has changed significantly, and now supports class abstractions, object-oriented programming, pure functions, and coarrays [1]. Coarrays have been part of the Fortran standard since Fortran 2008 and provide a syntactic extension of Fortran to support parallel programming; in the sequel we will use Fortran instead of CAF to highlight the fact that today the parallelization is an integral part of the language. Following the Partitioned Global Address Space programming model, in Fortran the program is replicated among a certain number of images, executing asynchronously. The PGAS model combines elgance and simplicity, by allowing one-sided communications, and potentially reducing synchronization and code complexity. Coarrays are a language-based approach, meaning that a quality compiler implementation can consider both serial and communication aspects in optimizing the generated code. If the available compiler also provides support for accelerator programming, e.g. by implementing OpenAcc, the programmer can readily get a dual benefit from the combined use of PGAS and accelerator programming models.

The Message Passing Interface (MPI) defines a suite of functions for communication exchange between processes. MPI is library-based: it is in principle independent from any specific compiler, but the user is bound to the vendor's fine-tuning of the MPI library implementation [2].