Solving Sparse Linear Systems of Equations using CAF

Ambra Abdullahi Hassan [a] Valeria Cardellini [a] and Salvatore Filippone [b]

[a] *University of Rome Tor Vergata, Rome, Italy*
[b] *Cranfield University, Cranfield, UK*

Abstract. Coarray Fortran (CAF) provides a syntactic extension to easily support parallel programming and has been incorporated into the standard since Fortran 2008. In this paper, we will investigate the migration from MPI to CAF of two libraries (PSBLAS and MLD2P4) for the solution of large linear systems of equations using iterative methods and preconditioners. Three main communication patterns have been identified: halo exchange, collective subroutines, and point-to-point communication. Furthermore, since PSBLAS and MLD2P4 implement a great number of methods and solvers, we aim at exploring different communication patterns involved in the solution of linear systems.

We will describe both performance studies and software quality analysis: the preliminary experiments we have conducted for the halo exchange in PSBLAS and using the GNU compiler and the OpenCoarrays library, show that the performances of the two CAF and MPI based implementations are comparable. With further tests we will compare performances of collective subroutines and point-to-point communication, and we will investigate whether a gain in performance can be achieved using other compilers. Additionally, since halo exchange, collective and point-to-point communications are all very common patterns in parallel software, we will describe the best-usage strategies for porting them from MPI to CAF.

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

1. Introduction

Fortran has been and still is a dominant language in High Performance Computing. Over the past decades Fortran has been greatly developed and now supports class abstraction, object-oriented programming, pure functions, and coarrays. Coarray Fortran (CAF) has been part of the Fortran standard since Fortran 2008 and provides a syntactic extension to support parallel programming. Coarrays are fully supported by Cray and Intel compilers. Although GNU compilers provide partial coarray support (only single image), the OpenCoarrays project provides two coarray transport layers, based on MPI and GASNet [1].

Following the Partitioned Global Address Space (PGAS) programming model, with CAF the program is replicated among a certain number of images, which execute asynchronously. A variable can be declared as a coarray, through the use of codimension and coindices: it implies that the variable is accessible from all the images. Combining elegancy and simplicity, CAF allows us to easily exploit one-sided communication, thus potentially reducing synchronization and code complexity. CAF is a language-based approach to parallelism, which means it is bound on specific compilers: when a good implementation is available, the compiler can concurrently consider serial aspects and communication for optimization. Additionally, if the compiler also supports OpenACC, the user can easily get a dual benefit from the combined use of CAF and GPUs.

MPI defines a suite of functions for communication exchange among processes. Differently from CAF, it is a library-based approach to parallelism: while independent from a specific compiler, the user is bound to the vendor's fine-tuning of the MPI library. Although MPI is the de facto standard for parallel programs running on distributed memory systems and so far few scientific softwares are written in CAF, it is a common feeling that many scientific applications can benefit from using CAF.

2. Contributions

The aim of this paper is to present the migration from MPI to CAF of two libraries (PSBLAS [2,3] and MLD2P4 [4]) for the solution of large systems of equations using iterative methods and preconditioners. Since the solution of large sparse linear systems is the most computationally expensive part of many scientific applications, studying the impact of CAF on the performance and software quality (not only in terms of coding but also of maintainability) on this kind of problems can be largely valuable.

PSBLAS [2,3] is a library of Basic Linear Algebra Subroutines that implements iterative solvers for sparse linear systems and includes subroutines for multiplying sparse matrices by dense matrices, solving sparse triangular systems, and preprocessing sparse matrices, as well as additional routines for dense matrix operations. MLD2P4 (Multi-Level Domain Decomposition Parallel Preconditioners) [4] is a package based on PSBLAS and implements various algebraic multi-level preconditioners. Both the libraries are implemented in Fortran 2003 and the current versions use message passing to address a distributed memory execution model.

We have converted the code gradually from MPI to coarrays, thus having MPI and CAF coexisting in the same code. We have identified three communication patterns that require modifications: the halo exchange, the collective subroutines, and the point-to-point communication. Halo exchange is performed every time a parallel matrix-vector multiplication is computed, and for this reason it is probably the most expensive communication in solving sparse linear systems. Since most of communication in MLD2P4 is based on PSBLAS routines, translation of MLD2P4 after PSBLAS comes with little effort. Several iterative solvers and preconditioners are implemented in PSBLAS and MLD2P4 using MPI. This allows us to explore different communication patterns involved in the solution of linear systems. For example, it is possible to compare performances of multigrid and domain decomposition methods in MPI and in CAF.

Few attempts to implement multigrid solvers in CAF have been made so far, e.g., in [5] where earlier versions of CAF and MPI have been used. In [6] a comparison between MPI3 and CAF is presented for spectral and multigrid techniques and for applications drawn from computational fluid dynamics. The work shows that the performance achieved by Cray implementations of MPI3 and CAF in message passing are comparable (with CAF slightly outperforming MPI3 in some cases), and it points out the advantages of CAF when it comes to ease of implementation.

We have conducted so far preliminary tests for the halo exchange in PSBLAS, using the GNU compiler and the OpenCoarrays library, showing that the performances achieved by the two CAF- and MPI-based implementations are comparable. With further tests we will study if some performance improvement can be achieved when translating to CAF collective subroutines and point-to-point communication, as well as whether any gain in performance can be achieved using CAF with other compilers. Since halo exchanges, collective subroutines, and point-to-point communications are all common pattern exchanges in parallel software, in addition to the performance studies and the software quality analysis, we also describe the best-usage strategies we have implemented in PSBLAS and MLD2P4 during their porting from MPI to CAF.

References

[1] A. Fanfarillo, T. Burnus, V. Cardellini, S. Filippone, D. Nagle, and D. Rouson. OpenCoarrays: Open-source transport layers supporting coarray fortran compilers. In *Proc. of 8th Int'l Conf. on Partitioned Global Address Space Programming Models*, PGAS '14, pages 4:1–4:11. ACM, 2014.

[2] S. Filippone and M. Colajanni. PSBLAS: a library for parallel linear algebra computations on sparse matrices. *ACM Trans. Math. Softw.*, 26(4):527–550, December 2000.

[3] S. Filippone and A. Buttari. Object-oriented techniques for sparse matrix computations in Fortran 2003. *ACM Trans. Math. Softw.*, 38(4):23:1–23:20, August 2012.

[4] P. D'Ambra, D. di Serafino, and S. Filippone. MLD2P4: a package of parallel algebraic multilevel domain decomposition preconditioners in Fortran 95. *ACM Trans. Math. Softw.*, 37(3), 2010.

[5] R. W. Numrich, J. Reid, and K. Kim. *Writing a multigrid solver using Co-array Fortran*, pages 390–399. PARA '98. Springer Berlin Heidelberg, 1998.

[6] S. Garain, D. S. Balsara, and J. Reid. Comparing Coarray Fortran (CAF) with MPI for several structured mesh PDE applications. *J. Comput. Phys.*, 297(C):237–253, September 2015.