Leveraging OpenCoarrays to Support Coarray Fortran on IBM Power8E

Alessandro Fanfarillo, Damian Rouson
Sourcery Inc.
www.Sourceryinstitue.org

Executive Summary

We report on the experience of installing and using the OpenCoarrays Message Passing Interface (MPI) transport layer to support the compilation of coarray Fortran (CAF) programs on a single IBM Power node in the Cedes cloud.

We include a few comparisons between the performance a competent application developer would attain with two-sided MPI communication and the performance achieved with the newer MPI 3.0 one-sided communication that OpenCoarrays generates from CAF programs. We also demonstrate the OpenCoarrays support for the parallel collective subroutines that have been proposed for the upcoming Fortran 2015 standard.

This paper's results and results we report elsewhere raise an important question for compiler development teams:

- Do the potential performance benefits and profit incentives associated with hardwiring CAF support to a proprietary communication library outweigh the performance benefits and cost reductions afforded by maintaining the flexibility to link applications to different communication libraries on different platforms?

Our results also raise an important question for application developers:

- Do the potential performance benefits of embedding communication library calls directly into source code outweigh the costs of having to re-engineer the application's communication kernels as the communication library evolves or as better libraries become available?

We believe the answers to these questions point toward compiler and application designs analogous to what OpenCoarrays facilitates.

Introduction

OpenCoarrays

OpenCoarrays[1] is an open-source software project for developing, porting and tuning transport layers that support coarray Fortran compilers. We target compilers that conform to the coarray parallel programming feature set specified in the Fortran 2008 standard. We also support several features proposed for Fortran 2015 the draft Technical Specification TS18508 Additional Parallel Features in Fortran[2].

The OpenCoarrays project team aims to

- Jump-start and accelerate compiler support for the internationally standardized coarray Fortran parallel programming model,
- Ease application developers' adoption of coarray Fortran by providing user training in parallel programming in modern Fortran, and
- Assist application developers with parallelizing legacy serial codes and co-developing modern parallel codes.

In support of the item 1 above, the OpenCoarrays team adopted a permissive license that is attractive to corporations interested in either incorporating OpenCoarrays into their own compilers or programming to the OpenCoarrays Application Binary Interface (ABI).

1 http://www.opencoarrays.org/

2 http://isotc.iso.org/livelink/livelink?func=ll&objId=17181227&objAction=Open

Besides providing source code, OpenCoarrays also makes an important design statement that we hope will influence the development of CAF compilers industry-wide. Whereas current commercial CAF compilers tie the compiler implementation to a vendor-specific transport layer, OpenCoarrays demonstrates the viability of liberating the application developer to link to the transport layer that is best suited for robust and efficient execution of their specific application on their specific platform. The same design decision liberates compiler teams from the need for costly rewrites should it prove profitable to change transport layers in subsequent compiler versions. We provide a single ABI capable of driving an MPI transport layer or transport layers built atop any one of several other communication substrates, including for example, GASNet[3], ARMCI[4], and GASPI[5].

For present purposes, we demonstrate the default OpenCoarrays transport layer LIBCAF_MPI using MPICH 3.1.4[6] and we compile CAF source codes using the GNU Compiler Collection[7] (GCC) 5.1 Fortran compiler GFortran. GFortran has adopted OpenCoarrays to support compilation of all parallel CAF executable files. However, OpenCoarrays remains a separate project with its own BSD-style license.

A broader study of the performance of OpenCoarrays in a variety of benchmarks and application prototypes running on a variety of platforms can be found in our PGAS 2014 paper[8].

Machine description and limitations

The IBM Power machine used during the investigation was a single virtual machine provided by Cedes equipped with a Linux Ubuntu 14.10 operating system. The product name assigned by Cedes to such system is Linux on Power VMs - Linux on Power VM (One Time - Short Term). The virtual machine exposes to the operating system eight POWER8E cores , 2 GB of RAM and 20 GB of hard disk.

In comparing the CAF performance with that of embedding two-sided MPI communication into Fortran source codes, it is important to note that all runs executed on a single virtual machine without any network involvement. Such an architecture could produce performance results that are not repeatable because of the platform's virtualized nature. The availability of POWER hardware at no cost was the sole driver on our use virtualization. We have successfully executed OpenCoarrays on larger, distributed-memory non-POWER platforms, and we expect no difficulties in executing on larger POWER platforms.

Methodology

Setting up

The default IBM Power machine provided a very limited set of pre-installed software packages. After the installation of some useful packages needed by GCC (e.g., binutils, make, flex, bison, etc.), we built GCC 5.1 from source. Compiling and installing GCC 5.1 was effortless with no errors or unexpected behaviours.

Next we used GCC 5.1 to build MPICH-3.1.4 from source. Again we encountered no difficulties. Finally, a simple issuance of ``make" built the OpenCoarrays LIBCAF_MPI library from our Gothab repository, again with no difficulties.

3 http://gasnet.lbl.gov/

4 http://hpc.pnl.gov/armci/

5 http://www.gaspi.de/

6 https://www.mpich.org/

7 http://gcc.gnu.org/

8 http://www.opencoarrays.org/publications

Tests: purpose and description

Our tests had a twofold purpose:

- To verify the functioning of OpenCoarrays-enabled CAF codes on an IBM POWER machine;
- To compare the performance of OpenCoarrays-enabled CAF codes to the more common scenario of Fortran source codes containing raw, two-sided MPI calls.

We served the first purpose by executing CAF codes that use Fortran 2008 communication and synchronization to perform a collective communication task. As a bonus, our results also demonstrated the superior performance of the proposed Fortran 2015 cosum and cobroadcast collective subroutines relative to their aforementioned counterparts that were manually coded in Fortran 2008.

We served the second purpose by executing a straightforward bandwidth benchmark. The bandwidth test case analyses the performance of two-sided MPI communication and CAF (using one-sided MPI 3.0 communication under the hood) during the transfer of data (C double values). The test repeats the measurement 100 times for each block size and returns the average value.

Results

Collectives

This section shows the performance of manually coded Fortran 2008 collective procedures and the proposed Fortran 2015 intrinsic collective subroutines as enabled in GFortran by OpenCoarrays. Figure 1 depicts the performance of a

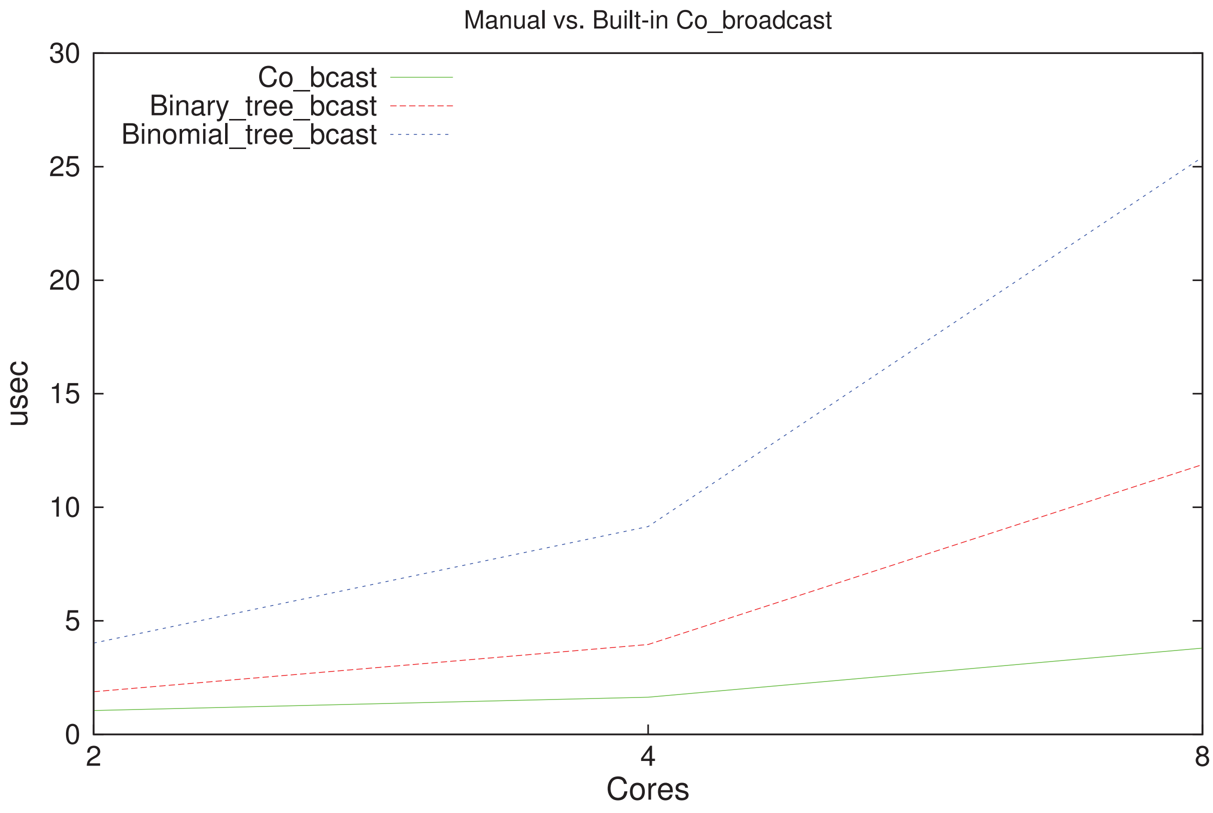

Figure 1: Comparison between proposed Fortran 2015 intrinsic broadcasts and manually constructed Fortran 2008 broadcast.

broadcast operation based on a binary tree approach transferring 32-bit integers written in Fortran 2008 and a broadcast using a binomial tree approach in Fortran 2008. Although the binary tree algorithm outperforms the binomial tree algorithm in the plotted results, tests we have run on other architectures indicate that the relative performance of the binomial and binary trees reverses when the number of cores involved increases.

In Figure 2 , we compare handwritten Fortran 2008 binary-tree, recursive-doubling, and alpha-tree algorithms with

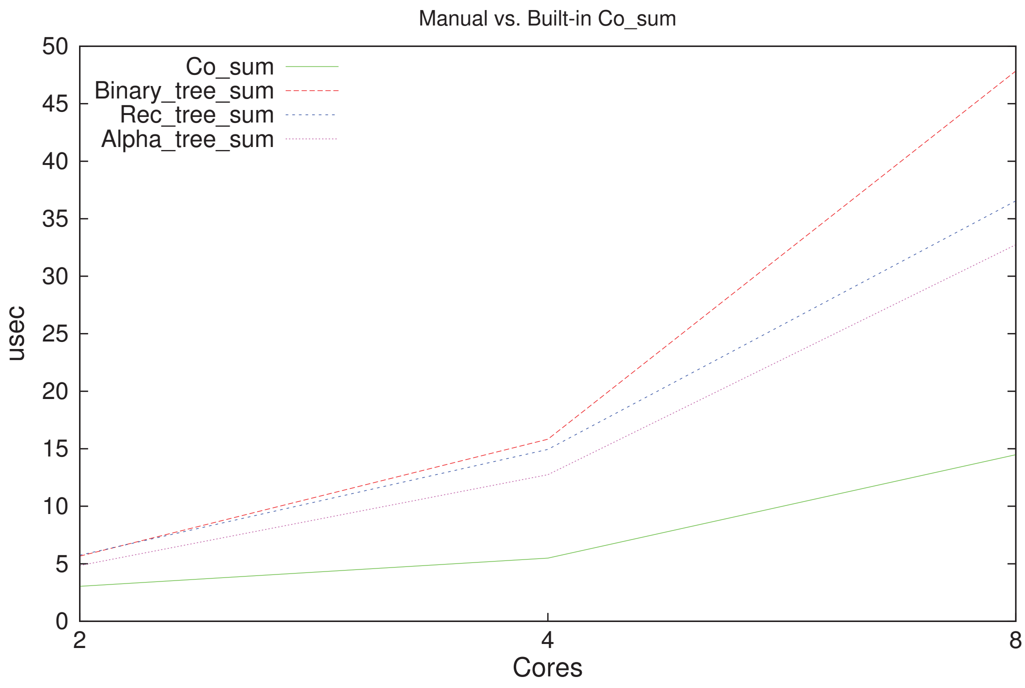

Figure 2: Comparison between proposed Fortran 2015 intrinsic collective summations and manually constructed Fortran 2008 collective summation.

OpenCoarrays Fortran 2015 cosum collective subroutine for parallel sum reduction. In investigations on larger platforms, we have observed similar trends for the relative performance of each handwritten algorithm with increasing core count.

Figures 1 and 2 exemplify the consistent superiority of the OpenCoarrays Fortran 2015 intrinsic collectives relative to a variety of handmade Fortran 2008 collectives. Source code for these tests will be incorporated into the test suite in the OpenCoarrays Gothab repository.

MPI vs. CAF

In this section, we compare the performance of two-sided MPI and CAF. The importance of this comparison lies in the potential performance benefits offered by the more advanced, one-sided MPI 3.0 communication that OpenCoarrays uses to support CAF codes. An added significance of the comparison lies in the increased complexity of one-sided MPI, which could prove prohibitive for many application developers.

Figure 3 shows, on logarithmic scale, that OpenCoarrays can outperform two-sided MPI in terms of bandwidth when

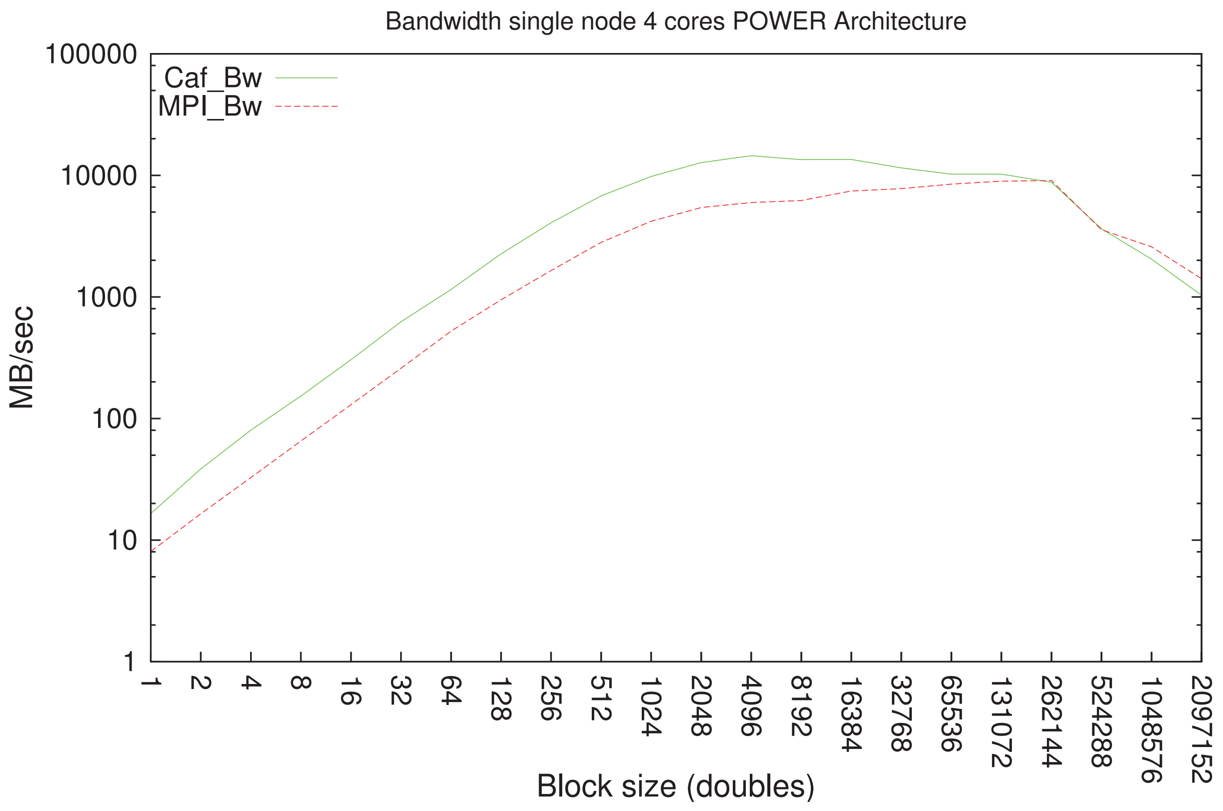

Figure 3: Bandwidth comparisons between MPI two-sided and coarrays

used on a single node. This advantage most likely stems from the efficient implementation of one-sided MPI provided by MPICH. Such a test case run in shared memory should not be interpreted as providing an exhaustive comparison between two-sided MPI and one-sided MPI-based CAF. With the latter caveat, the one-sided support of a modern MPI implementation compares favorably in this limited test with the two-sided approach found more commonly in applications.

Discussion

The results in Section embody the OpenCoarrays project team goal of ``jump-starting" development for both compiler teams and application developers. By providing early support for performance-enhancing Fortran 2015 features, OpenCoarrays enables compiler vendors and application developers to leapfrog the capabilities offered by current Fortran 2008 implementations. Furthermore, the communication bandwidth results demonstrate the competitiveness of the OpenCoarrays one-sided MPI communication model in comparison with the current widespread practice of embedding two-sided MPI in Fortran source codes.

Conclusions

Our results demonstrate the ease with which OpenCoarrays enables compiling CAF applications on an IBM Power machine. We employed GNU Fortran for convenience. We expect that linking OpenCoarrays to the proprietary IBM compiler and communication technologies would offer POWER platform-specific performance enhancement opportunities not contemplated or explored in the current study.

Our results also demonstrate the consistent superiority of the OpenCoarrays Fortran 2015 collective broadcast and summation subroutines relative to a variety of handwritten Fortran 2008 counterparts. Even though the limits imposed by access to a single compute node did allow for a larger-scale comparison, the results suggest that the one-sided MPI-based communication provided by OpenCoarrays, almost always outperforms MPI two-sided communication in simple data transfers.

Although we studied the behaviour of only the default OpenCoarrays LIBCAF_MPI transport layer, the favourable results suggest the viability of the OpenCoarrays design decision to liberate the compiler team and application developer from choosing a specific communication substrate. The results presented in this paper demonstrate that the resulting design can offer improvements in raw bandwidth as well as algorithmic improvements afforded by a rich set of intrinsic collective procedures. In previous tests not discussed in this paper, we have recompiled CAF codes without modification and linked them to other communion libraries such as GASNet. For some applications on some platforms, the resulting flexibility can offer additional performance enhancements not described in this paper. Such behaviour addresses the first of the two questions in the Executive Summary by indicating that the flexibility to vary the communication library supporting CAF offers a more attractive path than hardwiring a dependence on one communication library.

Regarding the second question in the Executive Summary, our experience suggests that many application developers are not keeping up with the evolution of the MPI standard, are not evaluating alternatives once a choice has been made, and are not confident in tackling the complexity of MPI 3.0 one-sided communication. In such cases, the performance benefits OpenCoarrays offers by insulating the application from MPI outweigh the performance costs of wrapping MPI. A CAF code supported by MPI 3.0 one-sided communication provided by OpenCoarrays outperforms a comparable Fortran code with raw, two-sided MPI embedded in the source.