Science & Technology Facilities Council

www.stfc.ac.uk

# Performance of Coarray Fortran vs MPI in a CFD Application

Mike Ashworth

Scientific Computing Department, STFC Daresbury Laboratory, Sci-Tech Daresbury, Warrington WA4 4AD, United Kingdom

## Introduction

Most scientific simulation codes running on distributed memory computers use a high-level language with MPI.

**MPI** is

- portable (easy to implement)
- simple (easy to learn)
- flexible (can implement any parallel algorithm)
- efficient (easy to optimize)

However, MPI codes can have many lines of additional message passing code and be difficult to maintain.

**Coarray Fortran** has several advantages [3]:

- Easy to write code; the compiler looks after the communication
- References to local data are obvious as such
- Easy to maintain code esp. more concise than MPI
- Integrated with Fortran for type checking, etc.
- The compiler can optimize communication
- Local optimizations are still available
- Few demands on the compiler, e.g. for coherency

## Shock-Boundary Layer Interaction (SBLI)

A finite difference formulation of Direct Numerical Simulation (DNS) of Turbulence from the University of Southampton [2].

A sophisticated DNS code that incorporates a number of advanced features:

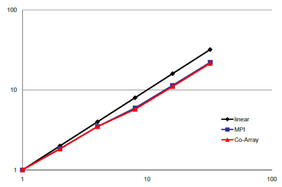
- High-order central differencing
- Shock-preserving advection scheme from the total variation diminishing (TVD) family
- Entropy splitting of the Euler terms
- Stable boundary scheme

Key computational features:

- Fortran 90 plus MPI
- 3D halo exchange
- With large domains scales to 168,000 cores

## Previous Work

*Figure 1. Speedup of the MPI and CAF codes relative to the MPI code on one processor of the Cray X1E*




Ashby and Reid [1] produced a CAF implementation of SBLI and tested it on the Cray X1E achieving identical performance to MPI. They concluded:

- The main advantage of using coarrays is that the code is more readable and understandable, and thus more maintainable
- Our tests show that the performance of our code does not suffer from the change to coarrays
- With the … Fortran 2008 standard, coarrays represent the first International Standard parallel programming language, … their use will enhance productivity and further the exploitation of parallelism

## The Coarray Implementation

Ashby and Reid handled variable array sizes by implementing arrays as allocatable components of coarrays of derived type.

Dynamic co-shaping, e.g. q[nx,ny,*], replaces the use of Cartesian communicators in the MPI version.

The 'sync all' statement is used for synchronisation.

Three options are available in the code:

**Option: Push vs. Pull**

- push (put) stores local data into memory on a remote process: a[k] = a
- pull (get) read data from a remote process: a = a[k]

**Option: Packing Data**

- packed: small transfers are packed into a buffer as in the MPI code
- non-packed: all transfers are expressed as separate coarray assignments no matter how small

**Option: Fortran 77 vs Fortran 90**

- Fortran 77: coarray assignments are made using nested do loops
- Fortran90: coarray assignments are expressed using Fortran 90 array syntax

## Performance

- Simple channel flow
- Domain of 120 x 120 x 120 run for 100 timesteps
- Run on ARCHER, the UK national supercomputer
- Cray XC30 node is 12-core Intel Ivybridge 2.7 GHz CPUs
- Cray PE 2.1.1, Cray Fortran 8.2.6
- Nodes linked using the Cray Aries interconnect
- Performance plotted is a constant (1000) divided by the execution time measured by system_clock

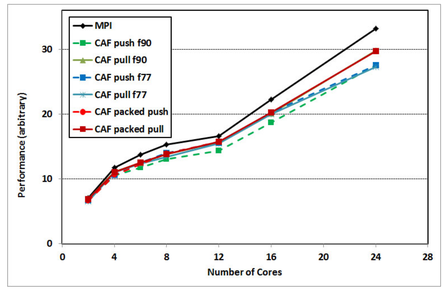



*Figure 2. Performance of MPI and various forms of CAF for SBLI up to 24 cores (one node) on the Cray XC30*

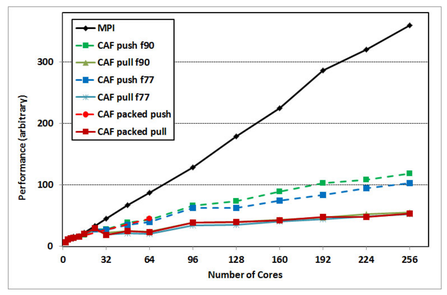



*Figure 3. Performance of MPI and various forms of CAF for SBLI up to 256 cores on the Cray XC30*

- CAF on a single node achieves up to 94% of MPI
- On multiple nodes this fraction ranges from 15%-33%
- Push (put) is better than pull (get)
- Packing into buffers and the Fortran 90 array syntax make little difference to performance

## Profiling

```
 Samp% |  Samp |  Imb. |  Imb. |Group
       |       |  Samp | Samp% | Function
       |       |       |       |  PE=HIDE
 100.0% | 731.2 |    -- |    -- |Total
|-------------------------------------------
|  64.4% | 471.2 |    -- |    -- |USER
||------------------------------------------
||  29.6% | 216.2 |  36.8 | 14.7% |rhs_
||   8.3% |  60.7 |  19.3 | 24.3% |ent_euler_
||   6.6% |  48.4 |  21.6 | 31.2% |d1x1_2Sderiv_
||   6.4% |  46.5 |  22.5 | 32.9% |dleta_2Sderiv_
||   5.6% |  41.1 |  17.9 | 30.6% |d1z_2Sderiv_
||   2.5% |  18.6 |  15.4 | 45.7% |d2x1_3Sderiv_
||   1.8% |  13.1 |   8.9 | 40.8% |d2eta_3Sderiv_
||   1.6% |  11.5 |  11.5 | 50.3% |d2z_3Sderiv_
||   1.3% |   9.2 |  13.8 | 60.5% |pdns3d_
|=============================================
|  29.3% | 214.0 |    -- |    -- |MPI
||------------------------------------------
||  11.3% |  82.4 |  95.6 | 54.1% |mpi_comm_dup
||   9.3% |  68.3 | 222.7 | 77.1% |mpi_bcast
||   4.4% |  32.5 |  47.5 | 59.8% |MPI_WAITANY
||   3.0% |  22.0 | 178.0 | 89.5% |mpi_recv
|=============================================
|   6.3% |  45.9 |    -- |    -- |ETC
||------------------------------------------
||   3.9% |  28.7 |  16.3 | 36.5% |_RTOR_W_00
||   1.5% |  10.6 |   8.4 | 44.4% |_RTOR_00
```

```
 Samp% |  Samp |  Imb. |  Imb. |Group
       |       |  Samp | Samp% | Function
       |       |       |       |  PE=HIDE
       |       |       |       |   Thread=HIDE
 100.0% | 3045.3 |    -- |    -- |Total
|-------------------------------------------
|  63.8% | 1943.3 |    -- |    -- |USER
||------------------------------------------
||  14.8% |  451.4 | 198.6 | 30.8% |pdns3d_
||   8.8% |  267.0 |  50.0 | 15.9% |rhs_
||   7.6% |  230.5 | 119.5 | 34.4% |swap_z_4_
||   5.3% |  162.1 |  66.9 | 21.9% |dist_xy_
||   3.6% |  108.7 |  92.3 | 46.3% |swap_y_4_
||   3.4% |  103.9 |  87.1 | 46.0% |period_xmb_
||   3.3% |  101.1 |  26.9 | 21.2% |swap_x_4_
||   2.6% |   79.1 |  95.9 | 55.2% |period_z_4_
||   2.0% |   61.9 |  20.1 | 24.7% |ent_euler_
||   1.7% |   50.6 |  21.6 | 30.3% |d1x1_2Sderiv_
||   1.6% |   48.6 |  20.4 | 29.8% |dleta_2Sderiv_
||   1.3% |   39.2 |  21.8 | 36.1% |d1z_2Sderiv_
||   1.0% |   31.0 | 153.0 | 83.8% |period_x_
|=============================================
|  36.0% | 1095.1 |    -- |    -- |ETC
||------------------------------------------
||  28.3% |  860.8 | 648.2 | 43.3% |_STOP2
||   6.1% |  186.8 | 243.2 | 57.0% |_pmi_smp_barrier_join
```

*Figure 4. CrayPAT output for the SBLI MPI version (left) and for CAF push f90 code (right) on 128 cores on the Cray XC30*

Profiling of both MPI and CAF code shows:

- We use non-blocking MPI calls so overheads are in MPI_WaitAny
- Equivalent wait times for CAF are in STOP2 in the ETC group
- Many of the user subroutines show a large increase in runtime
- Especially the main program pdns3d_ which increases from 9 to 452 samples
- CrayPat shows that the high water mark for memory usage for MPI is 62 MB but for CAF it is 102 MB

Many of these features are not understood and investigations are ongoing.

## References

[1] Ashby, J.V. and Reid, J.K., Migrating a Scientific Application from MPI to Coarrays, Proceedings of the Cray User Group Conference, 2008

[2] Redford, J.A., Sandham, N.D. and Roberts, G.T., Direct numerical simulation of transitional flow at high Mach number coupled with a thermal wall model, Computers & Fluids 45 pp. 37–46, 2011

[3] Reid, J.K., Parallel programming in Fortran with Coarrays, RAPS workshop, ECMWF, Reading, 5 November 2008

## Acknowledgement

This work is part of the Software Outlook program funded through a Service Level Agreement between STFC and the Engineering and Physical Sciences Research Council.