Reinventing Two-Sided Communication in Fortran to Transmit Polymorphic Objects Between Images

Brad Richardson

1 Motivation

As of the latest version of the Fortran standard ("Programming languages Fortran" 2023), there are a handful of constraints that prevent the effective use of polymorphic objects in coarrays. Specifically, it is not allowed to coindex an object which has any polymorphic components. There are reasons to want these restrictions for performance and memory management, but there are problems for which polymorphism is highly desirable. However, it is valid to use `co_broadcast` on objects with allocatable, polymorphic components. This paper describes a method of combining this feature with a novel use of the teams feature to re-derive a two-sided communication mechanism that is able to communicate polymorphic objects between images.

2 Usage

A variable of the derived type `communicator_t` defined in the `communicator` module is declared. The `init` type-bound procedure of this variable must be called by every image in the current team prior to initiating any communication. An image may then initiate transfer of any data object by calling the `send_to` type-bound procedure of that object and indicating the image to which the data should be sent. The identified image must execute a call to `receive_from` of the corresponding `communicator_t` variable identifying the image which is executing the `send_to` procedure. The `send_to` and `receive_from` procedures constitute synchronous, blocking operations. Thus, care must be taken to ensure that a deadlock does not occur. For instance, if all images initiate a send operation, then no image will be available to execute a corresponding receive, and all the images will wait forever.

An example program making use of the `communicator` library is shown below. The image with the largest index creates a message and sends it to its lower neighbor, i.e. image with one lower index. Each other image receives the message from its upper neighbor, i.e. image one higher index, and then sends it to its lower neighbor. Each image then prints the message.

```
program main
  use communicator, only: communicator_t

  implicit none

  type(communicator_t) :: comm
  character(len=20) :: message[*]
  integer :: me, ni
  class(*), allocatable :: payload

  call comm%init()
  me = this_image()
  ni = num_images()
  if (me == ni) then
      write(message, "(A,I0)") "Hello from image ", me
  else
      call comm%receive_from(me+1, payload)
      select type (payload)
      type is (character(len=*))
          message = payload
      class default
          message = "Didn't get a string message"
      end select
  end if
  if (me > 1) call comm%send_to(me-1, message)
  critical
      print *, "Received message '" // trim(message) // "' on image ", me
  end critical
end program
```

References

"Programming languages Fortran." 2023. Standard. Vol. 2023. Geneva, CH: International Organization for Standardization.