Fortran 2018: What's New

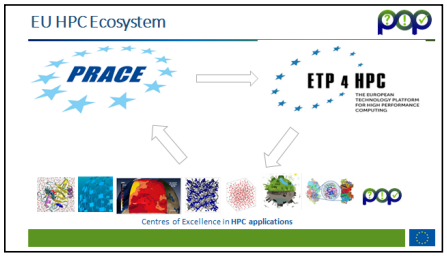
Research Software Engineers Workshop: London and South East

Numerical Algorithms Group
Wadud Miah
wadud.miah@nag.co.uk

nag®

Experts in numerical software and
High Performance Computing

POP CoE



- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **Services**
  - Precise understanding of parallel applications through parallel code profiling;
  - Suggestion/support on how to refactor code in the most productive way to increase parallel efficiency and scalability
- **Horizontal**
  - Transversal across application areas, platforms, scales
- **Free for academic, research AND commercial codes and users!**

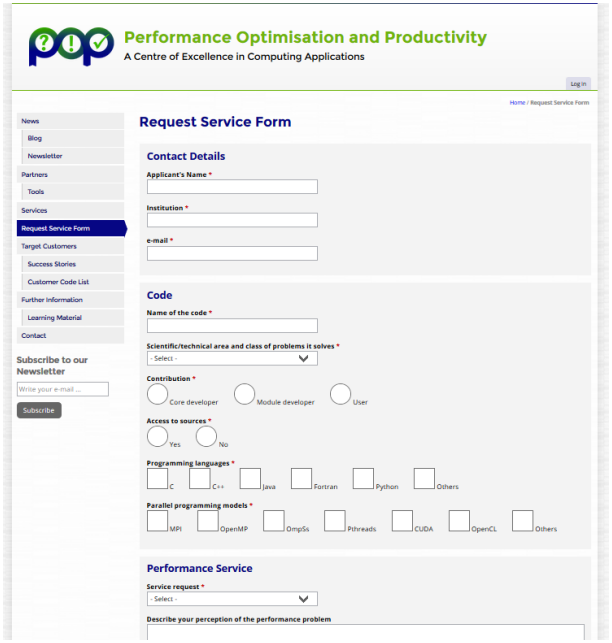
The process …

When?

- December 2018 - November 2021

How?

- Fill in small questionnaire describing application and needs https://pop-coe.eu/request-service-form
- Questions? Ask pop@bsc.es
- Install tools @ your production machine (local, PRACE, …)
- Interactively: Gather data → Analysis → Report
- Service is **free** for everyone!

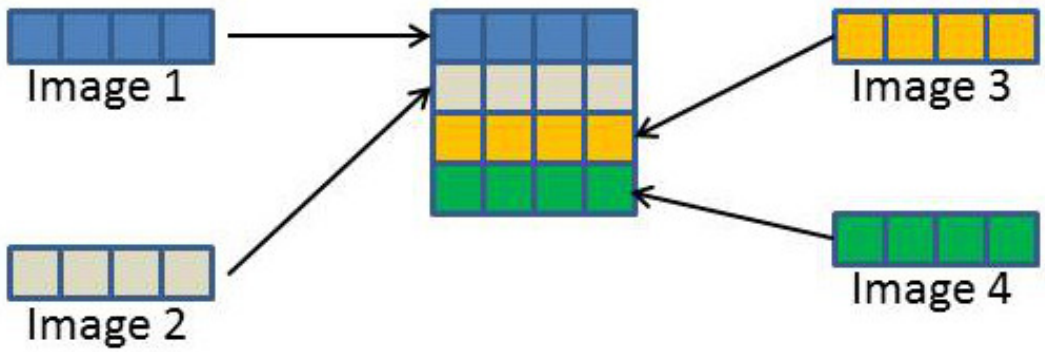


CoArrays - 2008

- *Shared* and *distributed* memory modes;
- Each process is called an *image* and communication between images is *single sided* and *asynchronous;*
- An image accesses remote data using CoArrays;
- Fortran is the only compiled language that provides distributed memory parallelism as part of the standard (Fortran 2008);
- *Supposed* to be interoperable with MPI;
- Coarrays have **corank**, **cobounds**, **coextent** and **coshape**. Indices used in coarrays are known as cosubscripts which maps to an image index.

CoArray Declaration (1)

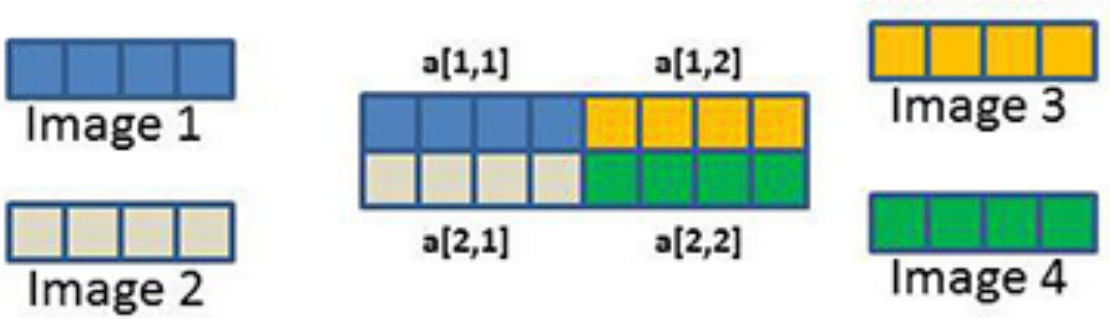
```
01 real, dimension(4), codimension[*] :: mat
$ aprun -n 4 ./caf_matrix.exe
```



- Coshape of coarray is `mat(:)[1:m]` where *m* is the number of images **which is specified at runtime**. In this example, it is 4;

CoArray Declaration (2)

```
01 real, dimension(4), codimension[2, *] :: mat
$ aprun -n 4 ./caf_matrix.exe
```



Fortran 2018 Collectives (1)

- New collective subroutines:

```
co_max( A [, result_image, stat, errmsg ] )
co_min( A [, result_image, stat, errmsg ] )
co_sum( A [, result_image, stat, errmsg ] )
```

- The above are collective calls and `A` must be the same shape and type;
- If `result_image` is supplied, it is returned to the specified image. It is undefined on all other images;
- `stat` and `errmsg` are returned and contain the status of the call;

Fortran 2018 Collectives (2)

- Broadcasts `a` from image `source_image` to all other images:

```
co_broadcast( a, source_image[, stat, errmsg ] )
```

- Reduction operation where `operation` is a pure function with exactly two arguments and the result is the same type as `A`:

```
co_reduce( a, operation[, result_image, stat, errmsg ] )
```

- If an image has failed, `stat=ierr` will be `STAT_FAILED_IMAGE`

CoArray Teams (1)

- Create new teams:

```
form team ( team_num, team_variable )
```

- `team_num` is an integer and `team_variable` is of `team_type`
- To change to another team:

```
change team ( new_team )
  ! statements executed with the new_team
end team
```

- Get the team number use `team_number( [ team ] )`

CoArray Teams (2)

- Below is an example taken from the 2018 standards document:

```
change team (team_surface_type)
  select case (team_number( ))
  case (LAND) ! compute fluxes over land surface
    call compute_fluxes_land(flux_mom, flux_sens, flux_lat)
  case (SEA) ! compute fluxes over sea surface
    call compute_fluxes_sea(flux_mom, flux_sens, flux_lat)
  case (ICE) ! compute fluxes over ice surface
    call compute_fluxes_ice(flux_mom, flux_sens, flux_lat)
  end select
end team
```

CoArray Teams (3)

- More intrinsic functions:

`this_image( team )` - returns the image index from `team`

`this_image( corray[, team] )` - returns a rank-one integer array holding the sequence of cosubscript values for coarray

`this_image( coarray, dim[, team] )` - returns the value of cosubscript `dim` in the sequence of cosubscript values for `coarray` that would specify an executing image, i.e. `this_image(coarray)[dim]`

`num_images( team )` - returns the number of images of `team`

`num_images( team_number )` - returns the number of images of `team_number`

Fortran 2018 Fault Tolerance

- Returns a list of images (integers of `KIND` type) that have failed or stopped:

```
failed_images( [ team, kind ] )
stopped_images( [ team, kind ] )
```

- The developer has to manually deal with image failures, e.g. read from the previous checkpoint and restart calculations;
- The argument `team` is of `team_type`;
- Returns `STAT_FAILED_IMAGE` or `STAT_STOPPED_IMAGE`:

```
image_status( image[, team] )
```

CoArrays Locks and Critical (1)

- Supports critical sections which can also be labelled:

```
UPDATE: critical
  i[1] = i[1] + 1
end critical UPDATE
```

- Supports locking to protect shared variables:

```
use iso_fortran_env
type(lock_type) :: lock_var[*]
lock( lock_var[1] )
i[1] = i[1] + 1
unlock( lock_var[1] )
```

CoArrays Locks and Critical (2)

- Can check to see if lock was acquired:

```
logical :: gotit

lock( lock_var[1], acquired_lock = gotit )
if ( gotit ) then
  ! I have the lock
else
  ! I do not have the lock - another image does
end if
```

Fortran Interoperability with C - 2003

- C is another major programming language in computational science and Fortran 2003 provides an interface to it;
- It uses the `iso_c_binding` intrinsic Fortran module;
- **Only assumed sized arrays are supported in 2008. Assumed shaped arrays are only supported in Fortran 2018**;

Fortran 2018 Interoperability with C

- Optional dummy arguments - `optional` attribute;
- Assumed-length character dummy arguments - `character(len=*), intent(in) :: header`
- Assumed shaped arrays - `real, intent(in) :: vec(:)`
- Allocatable dummy arguments - `real, allocatable, intent(out) :: table(:, :)`
- Pointer dummy arguments - `real, pointer, intent(in) :: vec(:)`

Optional Dummy Arguments (1)

- The optional argument is passed as a pointer to C. If the dummy argument is a NULL pointer, then it is not present;

```
subroutine print_header( debug )
  use iso_c_binding
  integer(C_INT), optional :: debug
  if ( present( debug )) then
    print '(I0,1X,A)', debug, 'Error found'
  else
    print '(1X,A)', 'Error found'
  end if
end subroutine
```

Optional Dummy Arguments (2)

- To call **with** the optional argument in the C code:

```
int debug = 4;
print_header( &debug );
```

- To call **without** the optional argument:

```
print_header ( (int *)0 );
```

Assumed-Length Character Dummy Arguments (1)

- Fortran calling C print function using descriptors:

```
interface
  subroutine print_header( msg ) bind(C)
    use iso_c_binding
    character(len=*,kind=c_char) , intent(in) :: msg
  end subroutine print_header
end interface
```

Assumed-Length Character Dummy Arguments (2)

```
#include <stdio.h>
#include "iso_fortran_binding.h"
void print_header( CFI_cdesc_t *msg ) {
  int ind;
  char *p = msg->base_addr;
  for ( ind = 0; ind < msg->elem_len; ind++ )
    putc( p[ind], stdout );
  putc( '\n', stdout );
}
```

C Descriptors (1)

- A C descriptor `CFI_cdesc_t` is a C structure with the following members:

`void *base_addr` - the address of the object. For unallocatable or disassociated pointers, it is NULL;

`size_t elem_len` - storage size in bytes;

`int version` - version number of the descriptor;

`CFI_attribute_t attribute` - whether the object is allocatable (`CFI_attribute_allocatable`), pointer (`CFI_attribute_pointer`) or neither (`CFI_attribute_other`).

`CFI_rank_t rank` - rank of the object and zero if a scalar;

C Descriptors (2)

`CFI_type_t type` - data type of this object. Macro can be `CFI_type_int`, `CFI_type_float`, `CFI_type_double`, `CFI_double_Complex`, and many other macros;

`CFI_dim_t dim[]` - describing the shape, bounds and memory layout of the array object;

- `CFI_index_t lower_bound` - the lower bound of array. Zero for everything else (member of `dim`);
- `CFI_index_t extent` - size of the dimension (member of `dim`);
- `CFI_index_t sm` - memory stride (member of `dim`).

C Example

```
void abs_array( CFI_cdesc_t *array )
  size_t i, nel = 1;
  for ( i = 0; i < array->rank; i++)
     nel = nel * array->dim[i].extent;

  if ( array->type == CFI_type_float ) {
    float *f = array->base_addr;
    for ( i = 0; i < nel; i++) f[i] = fabs( f[i] );
  } /* and for other real types */
}
```

Fortran Modernisation Workshop

- Two-day workshop covering modern Fortran, tools and libraries for computational science;
- Free for all, including academic, research and commercial;
- Workshops at ECMWF (Reading) between 1-2 April and Manchester University on 4-5 April 2019:

`www.nag.co.uk/content/fortran-modernization-workshop`

Let's Link Up

Ways to connect with us

Twitter: www.twitter.com/NAGTalk

Blog: http://www.nag.co.uk/blog

LinkedIn: http://www.linkedin.com/e/vgh/2707514/

nag®

Experts in numerical algorithms and HPC services