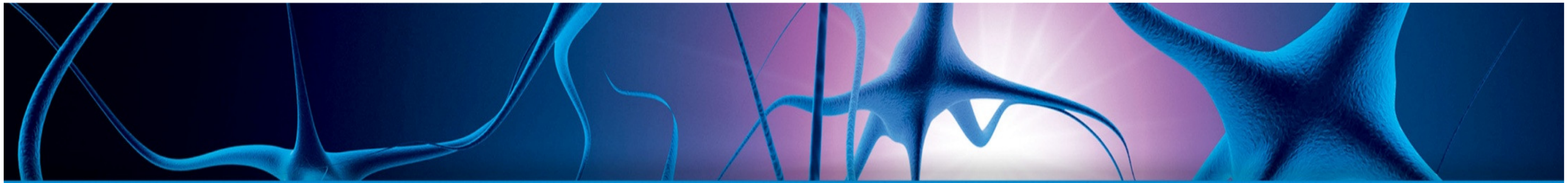


INTELLIGENT HPC: ILLUMINATE > INSIGHT > SOLVE > INTELLIGENT HPC: ILLUMINATE > INSIGHT > SOLVE >

Coarrays – A Parallel Programming Model in Intel Fortran

Steve Blair-Chappell
Software and Services Group
Intel Corporation

Sponsors of Tomorrow.™ intel®

What is Coarray Fortran?

- A parallel processing feature added to the Fortran language
- Part of the approved Fortran 2008 Standard
- A Partitioned Global Address Space (PGAS), Single-program Multiple-Data (SPMD) design
- Scalable from single-core to multi-CPU to clusters

History of Coarray Fortran

- Outlined in paper by Numrich and Reid in 1998
- Implemented by Cray for T3E and X-1
- Early preprocessor from Rice University
- Partial implementation in g95, experimental branch of gfortran
- Integrated into Fortran 2008 standard (approved in 2010)

Coarray Fortran Fundamentals: Images

- A CAF “Image” is a process
  - Processes have NO data sharing by default – separate memory maps.
- Example: hello world with no CAF syntax: 4 cores:

```
$> ifort -coarray -o hello hello.f90
$> ./hello
hello
hello
hello
hello
$>
```

```
program hello
write(*,*) ‘hello’
end program hello
```

```
program hello
write(*,*) ‘hello’
end program hello
```

```
program hello
write(*,*) ‘hello’
end program hello
```

```
program hello
write(*,*) ‘hello’
end program hello
```

CAF Fundamentals: Determining Number of Images, num_images()

- Intrinsic function num_images() returns an integer result, the total number of images in the CAF program:

```
$> cat hello_num_images.f90
program hello_num_images
  write(*,*) "Hello there are ", num_images()," total images"
end program hello_num_images

> ifort -coarray -coarray-num-procs=4 hello_num_images.f90
$> ./a.out
 Hello there are            4  total images
 Hello there are            4  total images
 Hello there are            4  total images
 Hello there are            4  total images
```

Coarray Fundamentals: this_image()

- Images have a logical ordering from 1 to N
- Integer function this_image() without an argument returns unique logical ordering from 1 to N
  - More complex image mappings possible: 2D, 3D, etc with arguments (topic discussed later)

```
$> cat hello_this.f90
program hello_this_image
  write(*,*) "Hello from image ", this_image()
end program hello_this_image
$> ifort -coarray -coarray-num-procs=4 hello_this.f90
$> ./a.out
Hello from image            1
Hello from image            3
Hello from image            2
Hello from image            4
```

- Remember, the images are inherently asynchronous

What is a coarray?

- Extends array syntax to add CODIMENSION
  - REAL, DIMENSION(100), CODIMENSION[*] :: X
  - REAL :: X(100)[*]
- Multiple codimensions possible
  - REAL :: X(100,200)[10,0:9,*]
- Scalars can also have codimensions
- Last bound of codimension is based on number of images
  - Last row may not be complete if images not a multiple of other codimension ranges
- Number of dimensions plus codimensions must be <= 15

What is a coarray? (contd.)

- Each copy of the program (image) has its own piece of the coarray
- References without [] mean local data
- References with [] mean data on specified image(s)
- Can use coarrays most places in the language
  - Coarrays may be allocatable, structure components, dummy or actual arguments

Where’s My Data?

REAL :: X(2,3)[*]

Image 1

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

REAL :: X(2,3)[*]

Image 2

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

REAL :: X(2,3)[*]

Image 3

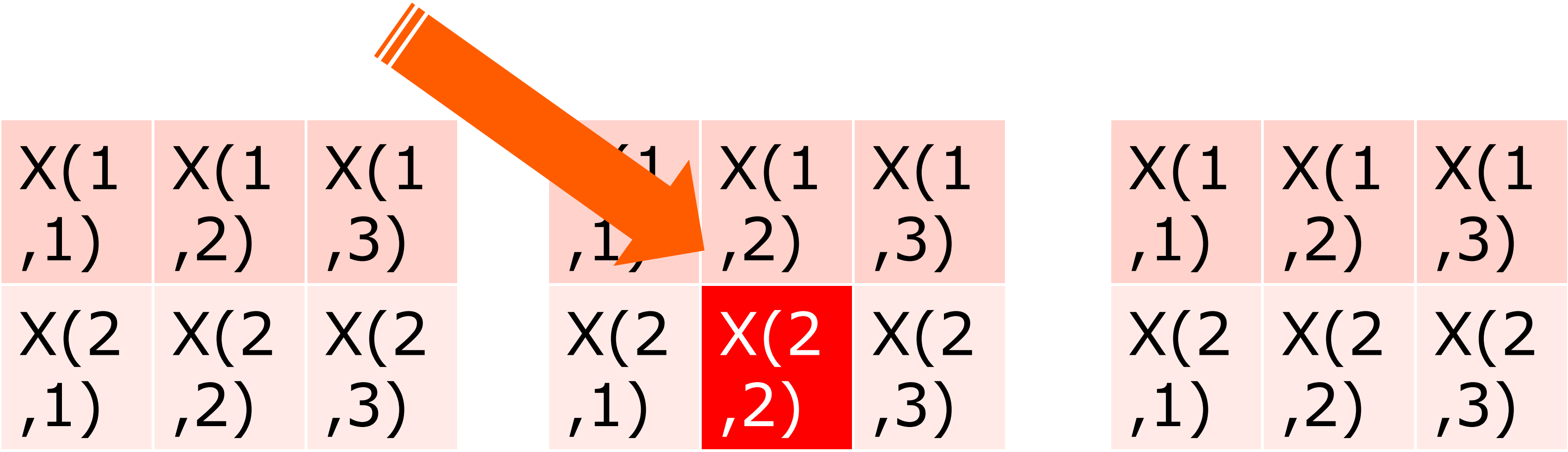
X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

Where’s My Data?

REAL :: X(2,3)[*]
Image 1

REAL :: X(2,3)[*]
Image 2

REAL :: X(2,3)[*]
Image 3



X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(2,2)[2] reference from image 1

Where’s My Data?

REAL :: X(2,3)[*]

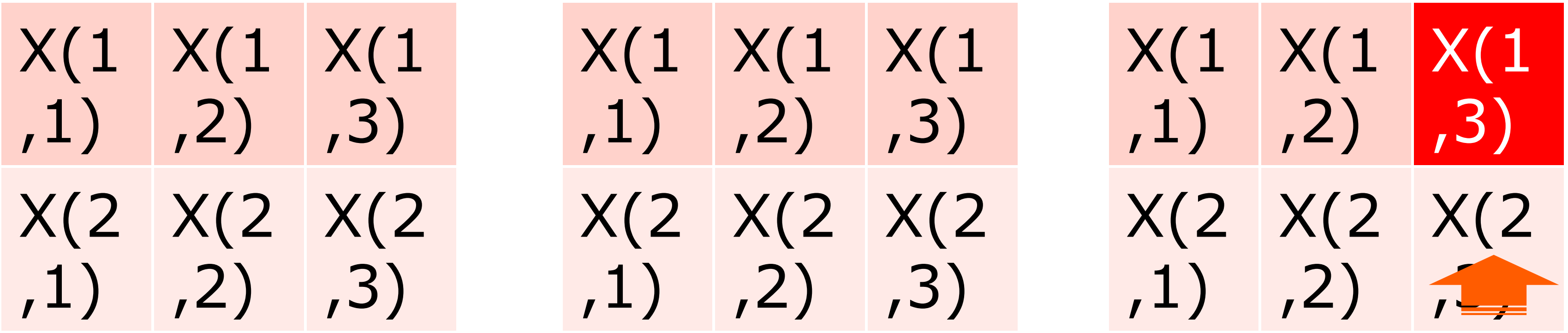
Image 1

REAL :: X(2,3)[*]

Image 2

REAL :: X(2,3)[*]

Image 3



X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(1,1)	X(1,2)	X(1,3)
X(2,1)	X(2,2)	X(2,3)

X(1,3) reference from image 3

Coindices

- Given `REAL :: Y[10,0:9,0:*], Z(10)[5,*]`
  - `Y[3,1,2]` accesses image 213
  - `Z(:)[1,4]` accesses image 16
- What if the specified image doesn't exist?

Error!

It's All About Image

- Number of images determined at run-time
  - Default is number of processor execution units
- `NUM_IMAGES` intrinsic tells you how many
- `THIS_IMAGE` intrinsic says which one you are
- `THIS_IMAGE(`*`coarray`*`)` gives you coindices for your copy of *coarray*
- `IMAGE_INDEX` converts coindices to image index

Staying in Synch

- `SYNC ALL`, `SYNC MEMORY`, `SYNC IMAGES` create synchronization points
- `CRITICAL/END CRITICAL` sections
- `LOCK` and `UNLOCK` statements control lock objects
- `ERROR STOP` terminates all images

More about Coarrays

- Each image has its own set of I/O units
  - “stdin” preconnected on image 1 only
  - “stdout” and “stderr” preconnected on all images
    - Implementation may merge them – not required
- Coarrays can be used in I/O
- Coarrays are not interoperable with C

Coarrays in Intel® Fortran

- Supported in Intel® Fortran Composer XE 2011 for Linux* and Intel® Visual Fortran Composer XE 2011 for Windows*
- Shared-memory implementation only in base product
- Distributed Memory implementation with addition of Intel® Cluster Toolkit license (Linux only at this time)

Coarrays in Intel Fortran

- Enable Coarray syntax with `–coarray (/Qcoarray` on Windows)
- Default number of images is same as number of processor execution units (processors*cores*threads)
  - Override with command option or environment variable
- `-coarray=distributed` to get distributed memory (cluster) – requires Cluster Toolkit license

Coarrays in Intel Fortran

- Underlying transport is Intel® MPI 4.0.1 for both shared and distributed memory
  - Other MPI implementations not supported
- At this time, **not** supported for use with OpenMP* or MPI direct calls
- With `–coarray=distributed`, uses existing configured MPI ring, or use `–coarray-config-file`

Running a Coarray Application

- For shared memory, just run it!
  - No mpirun, etc. needed – all handled automatically
- For distributed memory, need to start mpd first
- Environment variables available:
  - `FOR_COARRAY_CONFIG_FILE`
  - `FOR_COARRAY_NUM_IMAGES`

Example Program

```
if (this_image() == 1) print '(A,I0,A)', &
& "Coarray Fortran program running with ", &
& num_images(), " images"
sync all
print '(A,I0)',"Hello from image ", this_image()
end
```

Building and Running Example

```
c:\>ifort /nologo /Qcoarray caf.f90
c:\>caf.exe
Coarray Fortran program running with 8 images
Hello from image 1
Hello from image 5
Hello from image 2
Hello from image 3
Hello from image 7
Hello from image 4
Hello from image 6
Hello from image 8
```

Summary

- Single-Program-Multiple-Data (SPMD) model
- A fixed number of processes/threads called images all execute the same program asynchronously
- Coarray syntax specifies explicit data decomposition
- All data and computation is local to each image
- One-sided communication thru co-dimensions
- Explicit synchronization must be requested by programmer
- Supported by Intel® Fortran Compiler XE 2011 for Linux* and Windows* on IA-32 and Intel® 64 architectures

One More Thing…

There will be bugs…

Read the Release Notes for a list of known issues
Please let us know if you find others…

Optimization Notice

Intel® compilers, associated libraries and associated development tools may include or utilize options that optimize for instruction sets that are available in both Intel® and non-Intel microprocessors (for example SIMD instruction sets), but do not optimize equally for non-Intel microprocessors. In addition, certain compiler options for Intel compilers, including some that are not specific to Intel micro-architecture, are reserved for Intel microprocessors. For a detailed description of Intel compiler options, including the instruction sets and specific microprocessors they implicate, please refer to the “Intel® Compiler User and Reference Guides” under “Compiler Options." Many library routines that are part of Intel® compiler products are more highly optimized for Intel microprocessors than for other microprocessors. While the compilers and libraries in Intel® compiler products offer optimizations for both Intel and Intel-compatible microprocessors, depending on the options you select, your code and other factors, you likely will get extra performance on Intel microprocessors.

Intel® compilers, associated libraries and associated development tools may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include Intel® Streaming SIMD Extensions 2 (Intel® SSE2), Intel® Streaming SIMD Extensions 3 (Intel® SSE3), and Supplemental Streaming SIMD Extensions 3 (Intel® SSSE3) instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors.

While Intel believes our compilers and libraries are excellent choices to assist in obtaining the best performance on Intel® and non-Intel microprocessors, Intel recommends that you evaluate other compilers and libraries to determine which best meet your requirements. We hope to win your business by striving to offer the best performance of any compiler or library; please let us know if you find we do not.

Notice revision #20101101

Legal Disclaimer

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Performance tests and ratings are measured using specific computer systems and/or components and reflect the approximate performance of Intel products as measured by those tests. Any difference in system hardware or software design or configuration may affect actual performance. Buyers should consult other sources of information to evaluate the performance of systems or components they are considering purchasing. For more information on performance tests and on the performance of Intel products, reference www.intel.com/software/products.

Intel and the Intel logo are trademarks of Intel Corporation in the U.S. and other countries.

*Other names and brands may be claimed as the property of others.

http://intel.com/software/products

intel®

Software