
## Agenda

- How does Intel support coarray and will enhance it in the future?
- What is Intel thinking for support for many core CPU now and in the future?
- Where will Intel drive/take Fortran people to?
- What's New in Intel® Fortran 16.0
- Further  in future release (Fortran 17.0 later this year)

## How does Intel support coarray and will enhance it in the future?




## Intel Fortran's coarray support is based on Intel MPI

There are three driving forces behind coarray enhancements:

a. The draft Fortran 2015 Standard will have many new coarray features, like teams, atomics, collectives, events, and failed images, all based on the "almost finished" Technical Specification 18508. Intel Fortran will implement these coarray features plus the rest of Fortran 2015 in future releases.

b. Continuous improvement of coarray performance is essential to customers and the continued future success of coarrays. Much of that improvement will come in future releases from optimizing generated code that handles coarrays, like removing unnecessary locks, handling contiguous data in large chunks, and faster synchronization of images. Intel MPI currently supports the MPI 3 standard which contains features that may help us make other improvements in performance in future releases.

c. We are investigating the possibility that, in future releases, Intel Fortran could also support 3rd-party MPI implementations, like OpenMPI, MPICH, IBM MPI, or SGI MPI, as the base for coarrays.

What is Intel thinking for support for many core CPU now and in the future?




## Many-core CPU support

Intel Fortran will continue to support Intel's many-core, multi-core, and Xeon Phi CPUs through OpenMP constructs like TARGET and Intel directives like OFFLOAD.

Intel Fortran's parallel support is a hierarchy of features from finest to coarsest grain:

a. DO CONCURRENT from the Fortran 2008 Standard

b. Auto-parallel and PARALLEL, VECTOR, and SIMD directives

c. Coarrays

d. OpenMP

e. MPI

All of these features are areas for improvement in future releases.

## Where will Intel drive/take Fortran people to?




## Where to drive

Intel Fortran will continue to support new and emerging Fortran and OpenMP standards in future releases while supporting deprecated and deleted standard features and legacy features from VAX FORTRAN and Digital and Compaq Visual Fortran. Intel specific Fortran features may be added in future releases when hardware, operating system, or optimizer needs require them.

# What's New in Intel® Fortran 16.0




## New and Changed Features in Fortran 16.0

- Submodules features from Fortran 2008
- Further C Interoperability from Fortran 2015
- OpenMP 4.1 TARGET ENTER | EXIT DATA
- OpenMP 4.1 TARGET NOWAIT | DEPEND
- !DEC$ BLOCK_LOOP directive
- -init enhancements
- -fpp-name option
- VS2013 Shell

# Submodules (Fortran 2008)

The Problem:

- Any edit to a module, no matter how trivial, requires recompilation of all sources that USE that module, directly or indirectly
- Can cause a "recompilation cascade" in builds, greatly lengthening build time

The Solution:

- Submodules separate interface from implementation
- Changes in a submodule don't force recompile of module or sources that use the module (unless interfaces change)




# Further C Interoperability (Fortran 2015)

TS29113 on "Further Interoperability of Fortran with C" to be part of Fortran 2015. Motivations include:

- Support the needs of MPI3
- Provide Fortran equivalent of C's "`void*`" – assumed type and rank
- Enable C code to see array bounds, manipulate pointers and allocatables
- Extend interoperable interfaces to ALLOCATABLE, POINTER, assumed shape, CHARACTER(*) - all passed by new "C Descriptor"
- OPTIONAL allowed in interoperable interface
- Extend ASYNCHRONOUS beyond I/O
- Relax restrictions

# EXIT from BLOCK (Fortran 2008)

- When we first implemented BLOCK in Fortran 15, we didn't support EXIT from a BLOCK – now we do
- EXIT from other named constructs still in the future

```
outer: block
do i = 1, num_in_set
if ( x == a(i) ) exit outer
end do
call r
end block outer
```




# New and Changed Features in Fortran 16.0

- Submodules features from Fortran 2008
- Further C Interoperability from Fortran 2015
- OpenMP 4.1 TARGET ENTER | EXIT DATA
- OpenMP 4.1 TARGET NOWAIT | DEPEND
- !DEC$ BLOCK_LOOP directive
- -init enhancements
- -fpp-name option
- VS2013 Shell

Further in future release
(Fortran 17.0 later this year)




## Run-time performance improvements to coarrays

our improvements to coarray performance continue to focus on:

- transferring contiguous data in large chunks instead of element-at-a-time
- eliminating unnecessary LOCKs used for synchronization
- finding contiguous sub-slices of a non-contiguous coarrays to transfer
- using new features of the MPI3 standard in Intel MPI where we can, eg, one-sided communication

## For GET

- if b(i:j, k:l) is contiguous, the FRTL gets it from image 3 in one transfer; otherwise it gets each b(i:j, n) contiguous sub-slice

```
integer, allocatable :: a(:,:)[*], b(:,:)[*]
a(i:j,k:l) = b(i:j,k:l)[3]
```




## For PUT

- The optimization is not on the transfer of data but on the LOCK-UNLOCK around the transfer: one LOCK-UNLOCK surrounds all of the PUTs instead of one LOCK-UNLOCK for each PUT

```
outer: block
real, parameter :: init_value = huge()
real, allocatable :: rs(:,:)[*]

allocate(rs(100,100))
if (this_image == 1) then
    do i = 1, num_images()
        rs[i] = init_value
    end do
end if
```

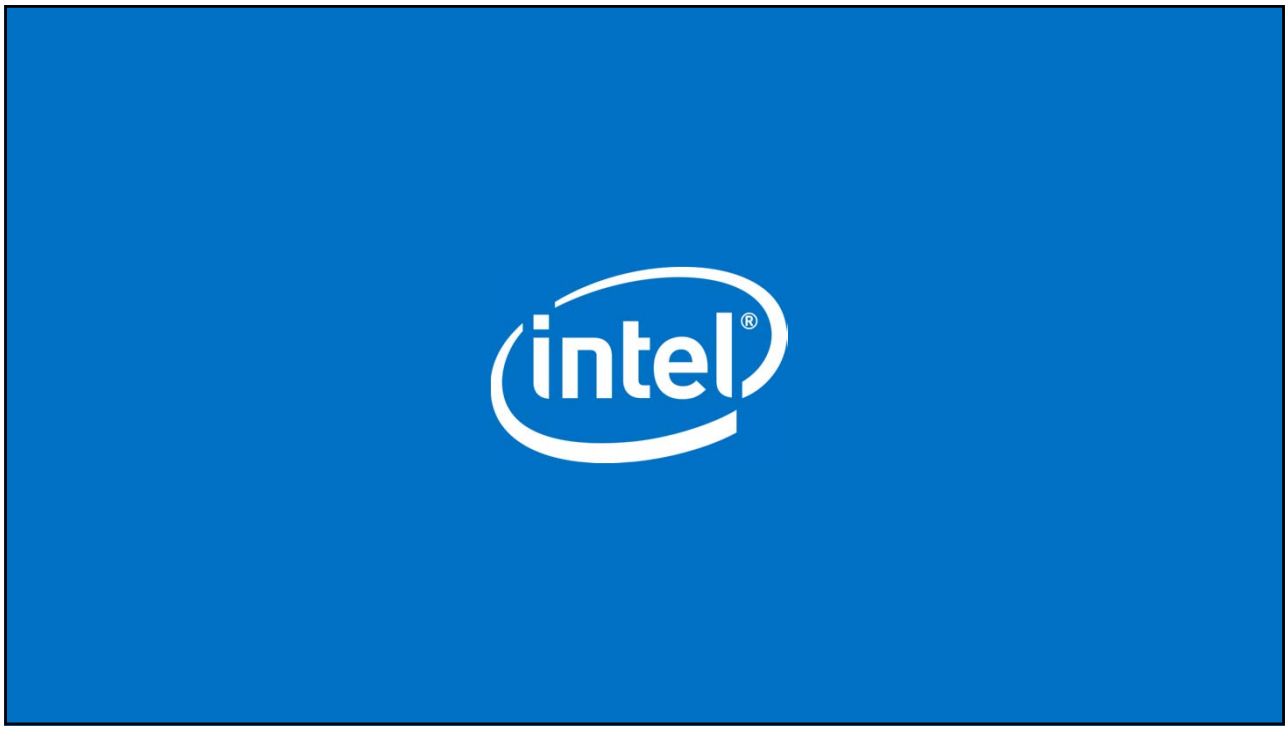
## Coarray performance improvements expectation – ifort 17.0 vs 16.0

- Small tests that ran in less than a second before, still run under a second, but are a little bit slower, due to increased overhead in RTL processing.
- On the other end of the scale, there were two tests that showed a 97+% improvement – Jacobi and a heat gradient test.
- Without including the numbers for Jacobi and heat, performance improves about 38%. With Jacobi and heat included, the overall improvement is 68%.




## Future releases of ifort will include new coarray features from the Fortran 2015 Standard

- Teams of images
- Failed image handling
- Event handling
- New atomic intrinsic procedures, eg, ATOMIC_ADD
- New collective intrinsic procedures, eg, CO_SUM




# Legal Disclaimer & Optimization Notice

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. **No computer system can be absolutely secure.** Check with your system manufacturer or retailer or learn more at **intel.com.**

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

The cost reduction scenarios described in this document are intended to enable you to get a better understanding of how the purchase of a given Intel product, combined with a number of situation-specific variables, might affect your future cost and savings. Nothing in this document should be interpreted as either a promise of or contract for a given level of costs.

Intel, the Intel logo, Xeon, Core, Iris Pro, and VTune are trademarks of Intel Corporation in the U.S. and other countries.
OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.

**Optimization Notice**

Intel's compilers 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 SSE2, SSE3, and 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. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804