INTEL® COMPILERS 2020 ROADMAP - TECHNOLOGY PREVIEW APRIL 2019

Intel Confidential

Content

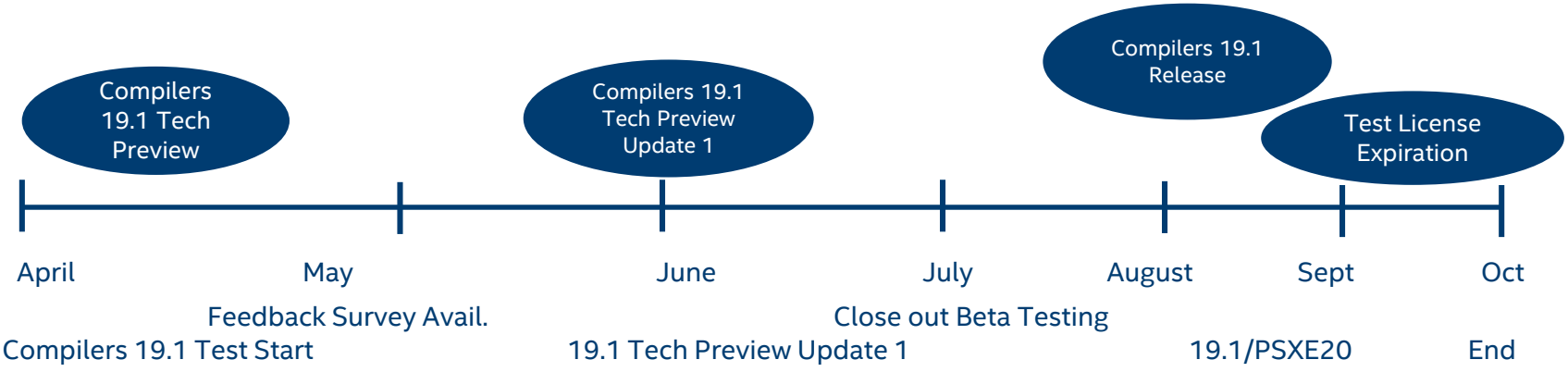
1) 2020 PSXE (compiler V19.1) Tech Preview Details
2) 2019 Roadmap/Timeline

PSXE 2020 Compilers v19.1 Tech Preview Details

- The next compiler version is **v19.1** BUT the package is Intel® Parallel Studio XE **2020**
  - A little confusing, apologies
- Compilers Tech Preview
  - Fortran v19.1: Broad customer testing for new Fortran features.
    1. A TON of new F2018 Standards features (Coarray Atomic, IEEE instrinsics, F18 Standards checking and compliance with –stand f18 see Backup slide)
  - C++: No broad public announcement
    - No major new features in ICC
    - Mostly bug fixes
- Feedback: End of Tech Preview Survey sent in early May, 2 weeks for user feedback

2019 Compilers Roadmap & Tech Preview Details

- 12 week testing program available for Tech Preview
- Compilers Tech Preview Features
  - Fortran v19.1: A TON of new F2018 Standards features (Coarray Atomic, Coarray CO_* routines, IEEE intrinsics, F18 Standards checking and compliance with –stand f18)
  - C++: No new features in ICC.



Timeline

- Target is 12 weeks for testing
- ~Mid-April Intel® Compilers v19.1 Technology Preview test start
- ~+1 Month later, open up the Survey for feedback
- ~Mid-June Update 1 releases (check for fixes, last chance for testing before final product drops)
- ~Mid July, close out testing
- End Q2/early Q3 Product Launch Intel Parallel Studio XE 2020 containing v19.1.0 compilers
- ~Mid October the test license expires

V19.1 Compiler OS Support - Windows

- Windows 10
- Windows Server 2019
- Windows Server 2016 (1607)
- Visual Studios Supported
  - Visual Studio 2017 w/ Windows SDK 10, VS2017 Build Tools w/ Windows SDK 10
  - Visual Studio 2019 w/ Windows SDK 10, VS2019 Build Tools w/Windows SDK 10
    - (Tech Preview Update 1 first support), no initial support in first Tech Preview release

Intel Visual Fortran Compiler Visual Studio Integration - Same as above VS IDEs

** Notes:

- No longer supplying VS Shell with Intel Visual Fortran,
- Support for VS 2015 dropped – Will not integrate into any VS older than 2017

V19.1 Compiler OS Support - Linux

Latest/Upcoming

- RHEL 7.5, RHEL8, expected Q2/Q3 2019
- SLES 15 to be released July 2019
- Fedora 28, Fedora 30 expected Q4 2018
- Ubuntu 18.04 LTS, 18.10, 19.04 expected Q2 2019
- Debian 9 released June 2017
- Amazon Linux
- Intel Clear Linux

NOTES: Distros based on above, ex CentOS OpenSuse, should work

Newer GCCs required, need to move to more modern Linux distros for 2020

V19.1 Compiler OS Support - macOS

macOS 10.14 Mojave, Xcode 10

Later in 2019 macOS 11.x, Xcode 11

* Starting with macOS 11, 32-bit no longer supported (per Apple support)

Questions?

- How do I sign up?
  - Watch Intel's Fortran User Forums
  - Once the Tech Preview opens ~April 9-15th:
    - Register to get a license (special license needed)
    - https://software.seek.intel.com/compilers-2020-preview
  - OTHER QUESTIONS?

Send email to ron.green@intel.com

Legal Disclaimer & Optimization Notice <w/o benchmarks>

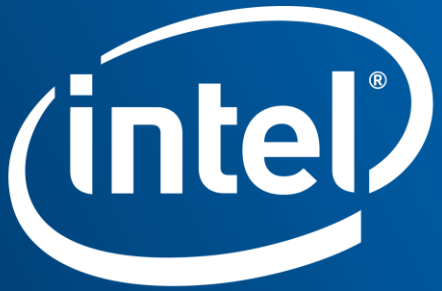
Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may 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. For more complete information visit www.intel.com/benchmarks.

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.

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



Software

BACKUP

What is a Technology Preview

- Like a Beta but …
  - In the Tech Preview package, **Only the Compilers will be new**
  - Less emails for you, the customer
  - Little/no questions to get the test license and download the package
  - Post-test Survey questions will also be short and simple
  - Will use our normal “Beta” processes
    - “Beta” license to obtain access
    - Intel Registration Center for downloading
    - Intel Online Service Center and User Forums to report bugs/issues

Fortran New Features – Big List

F2015: Implement SELECT RANK construct
F2015: An F2015 compiler is required to diagnose use of non-standard intrinsic procedures and modules, or non-standard intrinsics from standard modules
F2008/F2015: Transformational functions defined in ISO_C_BINDING, IEEE_ARITHMETIC, and IEEE_EXCEPTIONS are allowed in specification expressions
F2015: Issue warning that labeled DO loops are now obsolescent
F2015: The arithmetic IF statement has been deleted
F2015: COMMON, EQUIVALENCE, and BLOCK DATA are now obsolescent
F2015: Non block DO construct is a deleted feature
F2015: The SIZE= specifier can now be used with advancing input
F2015: The value assigned by the RECL= specifier in an INQUIRE statement now standardized
F2015: D, E, EN, and ES edit descriptors can have a field width of zero, analogous to the F edit descriptor
F2015: The exponent width e in a data edit descriptor may be zero, analogous to a field width of zero
F2015: Floating-point formatted input accepts hexadecimal-significand numbers conforming to ISO/IEC/IEEE 60559:2011
F2015: Implement the new EX edit descriptor
F2015: Reference to intrinsic CMPLX with a complex actual argument no keyword is needed for the KIND argument.
F2015: The arguments to the SIGN function may be of different kinds
F2015: A GENERIC statement can now be used to declare generic interfaces
F2015: IMPLICIT NONE (EXTERNAL | TYPE)
F2015: Specific names for intrinsic procedures are obsolescent
F2015: Implement new ATOMIC intrinsics
F2015: Implement coarray collective intrinsic procedures
F2015: Locality of variable in DO CONCURRENT can be declared on the DO CONCURRENT statement
F2018: Add optional ROUND argument to the IEEE_RINT function F2018: Implement the IEEE_NEXT_DOWN and IEEE_NEXT_UP intrinsic module functions
F2018: Add the IEEE_AWAY rounding mode
F2018: Implement the IEEE_FMA (fused mulitply-add function
F2018: Implement the IEEE_SIGNBIT intrinsic module function
F2018: Add optional RADIX argument to the IEEE_GET_ROUNDING_MODE and IEEE_SET_ROUNDING_MODE functions
F2015: Implement IEEE_MAX|MIN_NUM[_MAG] intrinsics
F2015: SUBNORMAL is now synonymous with DENORMAL
F2015: Implement IEEE_QUIET|SIGNALING_COMPARE where COMPARE is EQ, GE, GT, LE, LT, or NE.
F2015: Values for POS= and SIZE= in and INQUIRE statement for pending asynchronous operations have been standardized
F2015: add STAT arguments to ATOMIC_DEFINE and ATOMIC_REF, STAT and ERRMSG to MOVE_ALLOC and CRITICAL construct, and STAT= to image selectors

C++ New Features

- C++17 COMPLETE! Moving towards C++20
- C++20 features consteval and constexpr virtual functions coming Update1

```
consteval int sqr(int n) {
 return n*n;
 }

 constexpr int r = sqr(100); // OK

int x = 100;
int r2 = sqr(x);   // Error: Call does not produce a constant
```

```
consteval int f() { return 42; }
consteval auto g() { return f; }
consteval int h(int (*p)() = g()) { return p(); }
constexpr int r = h(); // OK

 constexpr auto e = g(); // ill-formed:
                         // a pointer to an immediate
                         // function is
                         // not a permitted result
                         // of a constant expression
```

Cascade Lake Compiler Support

- What's new: VNNI, Vector Neural Network Instructions
  - Designed for convolutional neural network (NN) based algorithms
- Compiler support via intrinsics and in-line assembly
  - Useful for creators of NN libraries, kernels (MKL-DNN for example)
  - .../include/icc/zmmintrin.h v19 and later compilers
- NO compiler auto-generation of VNNI instructions
  - No way to recognize high-level algorithms and translate to VNNI
- -x and –ax cascadelake compiler options for tuning (no –m option)
  - Same as –xskylake-avx512 –qopt-zmm-usage=high