Introducing Intel oneAPI 2023

oneAPI – 가속 컴퓨팅을 개발하기 위한 스마트한 방식

2023. 04. 19.
MOASYS

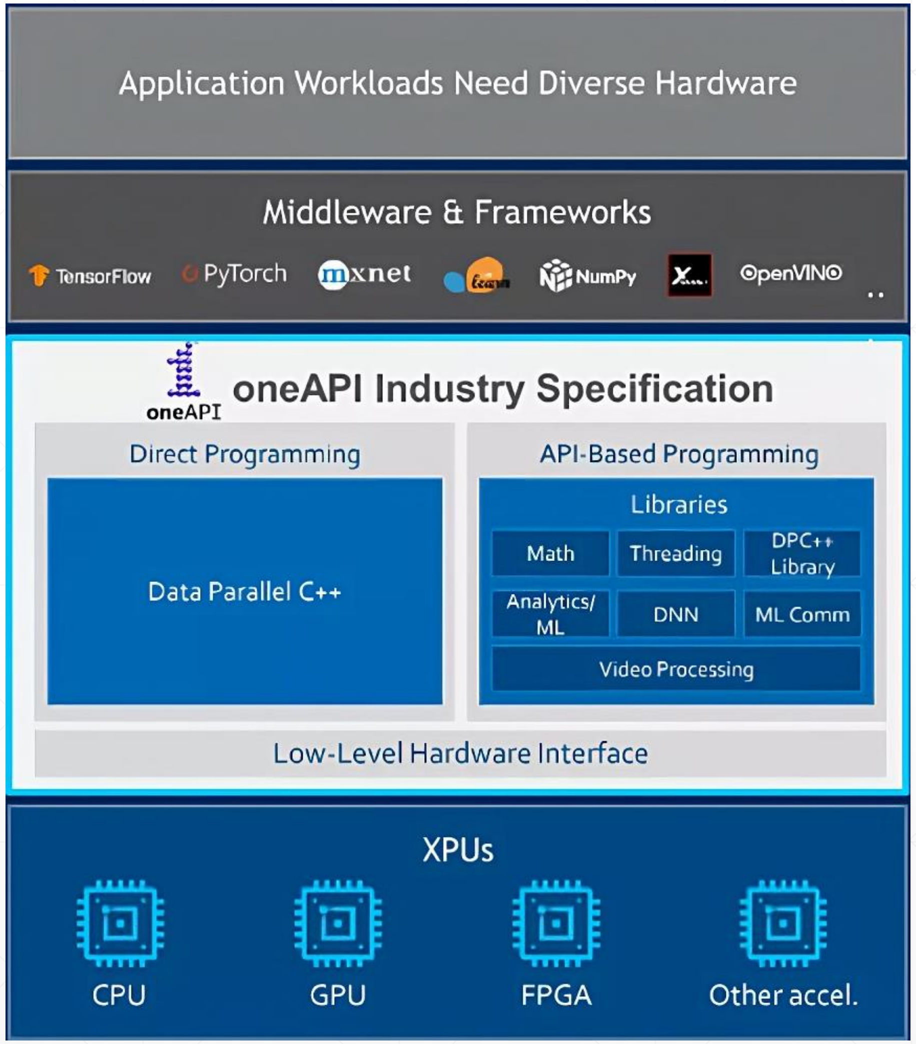
목차

아키텍처 간 가속 프로그래밍을 위한 스마트한 방식

- 최선의 선택
  - 여러 아키텍처와 벤더를 지원하는 하나의 프로그래밍 모델
  - 크로스 아키텍처 코드 재사용으로 벤더 종속성에서 자유로움

- 하드웨어 가치 극대화
  - CPU, GPU, FPGA 및 기타 가속기로 뛰어난 성능 제공
  - 최신 하드웨어의 최첨단 기능 활용 가능

- 업계 표준의 소프트웨어 개발 및 배포
  - C, C++ with SYCL, Python, OpenMP, Fortran, MPI 등 기존 언어 및 프로그래밍 모델과 호환
  - 도메인 별 기능을 가속화하는 강력한 라이브러리 제공



Intel® oneAPI Toolkit

Intel® oneAPI Base Toolkit

Native Code Developers

A core set of high-performance tools for building C++, Data Parallel C++ applications & oneAPI library-based applications

Add-on Domain-specific Toolkits

Specialized Workloads

Intel® oneAPI Tools for HPC

Deliver fast Fortran, OpenMP & MPI applications that scale

Intel® oneAPI Tools for IoT

Build efficient, reliable solutions that run at network's edge

Intel® oneAPI Rendering Toolkit

Create performant, high-fidelity visualization applications

Toolkits powered by oneAPI

Data Scientists & AI Developers

Intel® AI Analytics Toolkit

Accelerate machine learning & data science pipelines with optimized DL frameworks & high-performing Python libraries

Intel® Distribution of OpenVINO™ Toolkit

Deploy high performance inference & applications from edge to cloud

oneAPI의 생태계

End Users

Accrad, archanan, kt, FOUNDRY. MODO, AUTODESK ARNOLD, BioDataAnalysis GmbH, GeoEast, 中国石油集团东方地球物理勘探有限责任公司 BGP INC. CHINA NATIONAL PETROLEUM CORPORATION, BENTLEY, CINESITE, LAIKA, EURECOM, Brightskies, Verizon, PHILIPS, 大势智慧 DASPATIAL, WeBank 微众银行, SAMSUNG MEDISON, TANGENT ANIMATION, allegro.ai, ILLUMINATION MACGUFF

Institutes

Argonne NATIONAL LABORATORY, SiPearl, CERN openlab, CINECA, UT-BATTELLE Oak Ridge National Laboratory, Peraton Labs, lrz Leibniz Supercomputing Centre of the Bavarian Academy of Sciences and Humanities, SANKHYA SUTRA, Laboratório Nacional de Computação Científica

Vendors

codeplay, CANONICAL, CERTIFACE, ANACONDA, Ansys, MAX PLANCK COMPUTING & DATA FACILITY, MPCDF, AI SINGAPORE, sas, Red Hat OpenShift Data Science, KATANA GRAPH, CGG, AIBLE, CHAOSGROUP, MAXON, Ospirent, FOUNDRY., mercenaries engineering Guerilla, SENAI CIMATEC FIEB, SAP, E4 COMPUTER ENGINEERING, YUAN, GIGASPACES innovate with confidence, MEGH COMPUTING, KFBIO, Tech Mahindra, AsiaInfo 亚信科技, SUSE, UNITED IMAGING

OEM

BittWare a molex company, Hewlett Packard Enterprise, DELL Technologies, Atos, Lenovo, HCL, ENIAC, MEGWARE

Universities

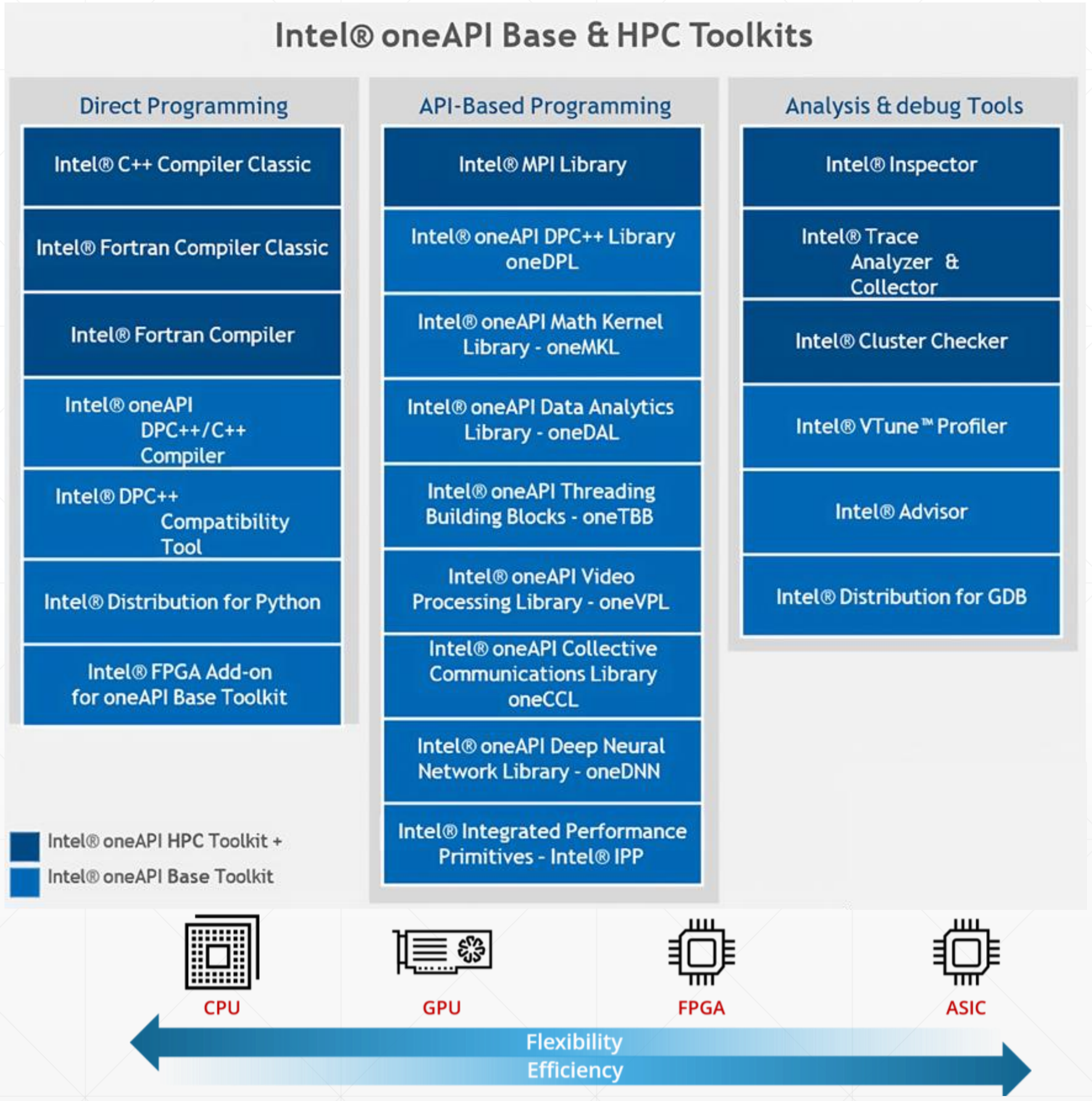
Ben-Gurion University of the Negev, Berkeley, UC DAVIS UNIVERSITY OF CALIFORNIA, LOBACHEVSKY UNIVERSITY, THE UNIVERSITY OF TENNESSEE KNOXVILLE, UNIVERSIDAD COMPLUTENSE MADRID, TECHNION Israel Institute of Technology, OLD DOMINION UNIVERSITY, UNIVERSITY OF CAMBRIDGE, ctg, ICT 中国科学院计算技术研究所 INSTITUTE OF COMPUTING TECHNOLOGY, CHINESE ACADEMY OF SCIENCES, 北京大学软件与微电子学院 School of Software & Microelectronics, ZIB, ILLINOIS, THE UNIVERSITY OF UTAH, University of Stuttgart Germany, FACULTY OF MATHEMATICS AND PHYSICS Charles University, UNIVERSITY OF OREGON, TÉCNICO LISBOA, inesc id lisboa, University College London, Durham University, सी डैक CDAC, UKRI Science and Technology Facilities Council Scientific Computing, TACC, Indian Institutes of Technology Delhi / Kharagpur / Roorkee, SDSC SAN DIEGO SUPERCOMPUTER CENTER, UNIVERSIDAD DE MÁLAGA, URZ HEIDELBERG UNIVERSITY COMPUTING CENTRE, NIU Northern Illinois University, University of BRISTOL, Stockholm University, KTH, Indian Institute of Science Bangalore, IISER PUNE Indian Institute of Science Education & Research Pune

ML/AI Frameworks

Microsoft Azure, Alibaba Cloud, Google Cloud, TensorFlow, Taboola, Tencent 腾讯, DataRobot, Bai百度, 飞桨, NAVER CLOVA, PyTorch

Intel® oneAPI BASE Toolkit: 아키텍처 간 개발 간소화

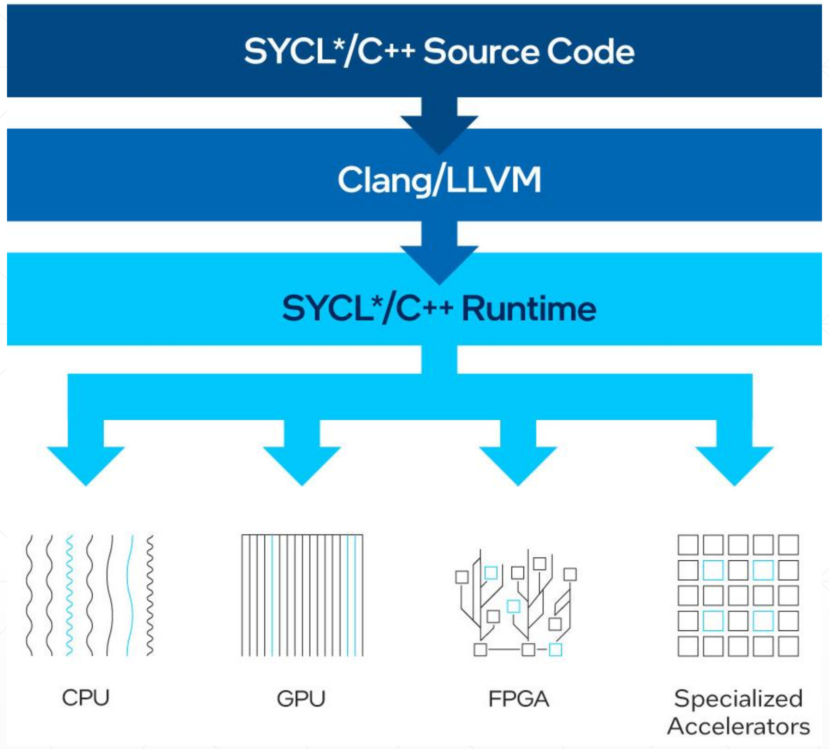
- 데이터 병렬 C++ 컴파일러, 라이브러리 및 분석 도구
- Intel® DPC++ 호환성 도구
- 스레딩, 수학, 데이터 분석, 딥러닝, 비디오/이미지/신호 처리에 최적화된 성능 라이브러리
- Intel® VTune™ 프로파일러로 병목 구간을 분석하여 코드 성능 최적화
- Intel® AI 도구으로 주요 DL 및 ML 프레임워크 지원
- 최적화된 scikit-learn, NumPy, SciPy 라이브러리가 포함된 Python 배포판



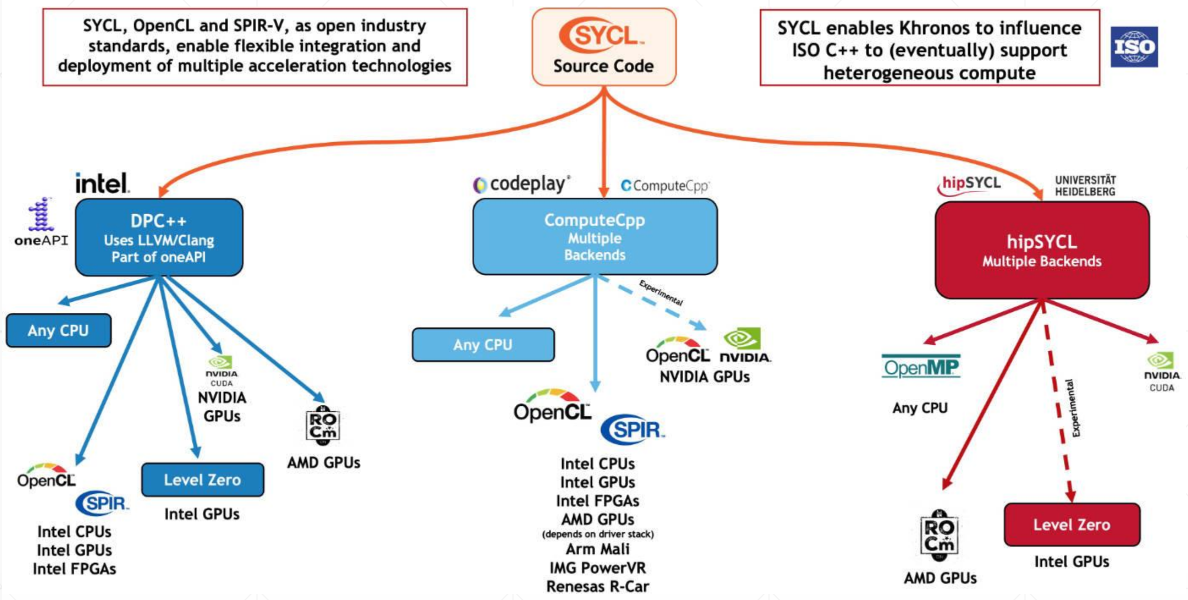
Intel® oneAPI Base & HPC Toolkits
Direct Programming
Intel® C++ Compiler Classic
Intel® Fortran Compiler Classic
Intel® Fortran Compiler
Intel® oneAPI DPC++/C++ Compiler
Intel® DPC++ Compatibility Tool
Intel® Distribution for Python
Intel® FPGA Add-on for oneAPI Base Toolkit
API-Based Programming
Intel® MPI Library
Intel® oneAPI DPC++ Library oneDPL
Intel® oneAPI Math Kernel Library - oneMKL
Intel® oneAPI Data Analytics Library - oneDAL
Intel® oneAPI Threading Building Blocks - oneTBB
Intel® oneAPI Video Processing Library - oneVPL
Intel® oneAPI Collective Communications Library oneCCL
Intel® oneAPI Deep Neural Network Library - oneDNN
Intel® Integrated Performance Primitives - Intel® IPP
Analysis & debug Tools
Intel® Inspector
Intel® Trace Analyzer & Collector
Intel® Cluster Checker
Intel® VTune™ Profiler
Intel® Advisor
Intel® Distribution for GDB
Intel® oneAPI HPC Toolkit +
Intel® oneAPI Base Toolkit
CPU
GPU
FPGA
ASIC
Flexibility
Efficiency

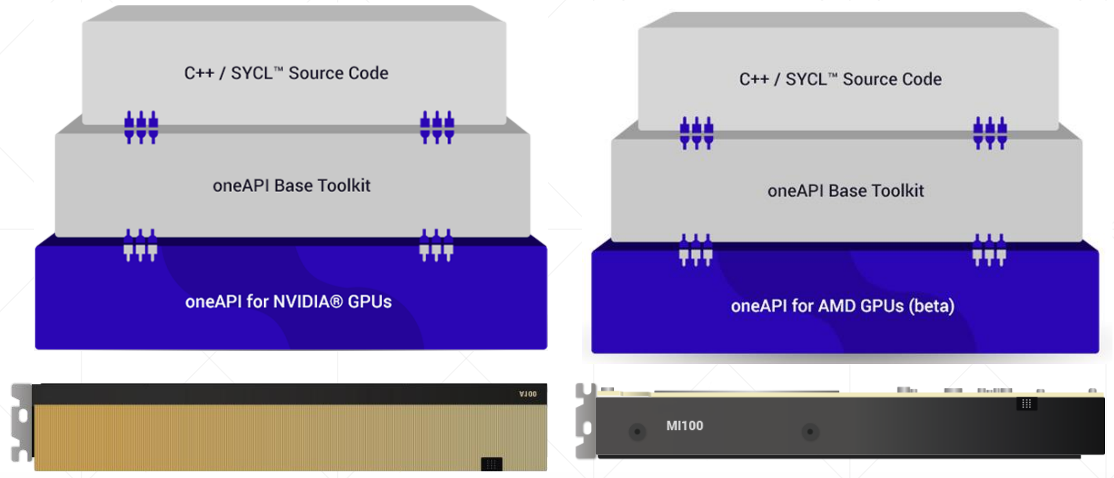
Data Parallel C++: 개방적인 크로스 아키텍처 프로그래밍 솔루션

- DPC++ = [C++] + [SYCL] + [확장성]
- CPU와 가속기에 대한 개방적인 병렬 프로그래밍 제공
  - 타겟 하드웨어 간에 코드 재사용이 가능하며 특정 액셀러레이터용 튜닝이 가능
  - 단일 아키텍처 전용 언어를 대체하는 개방형 업계 전체의 대체 수단
- Khronos SYCL 표준
  - C++의 생산성 장점 제공
  - 익숙한 C 및 C++ 구조를 사용하여 데이터 병렬 처리와 이기종 프로그래밍을 지원
- 인텔의 수십 년간의 아키텍처와 고성능 컴파일러 경험을 기반으로 구축



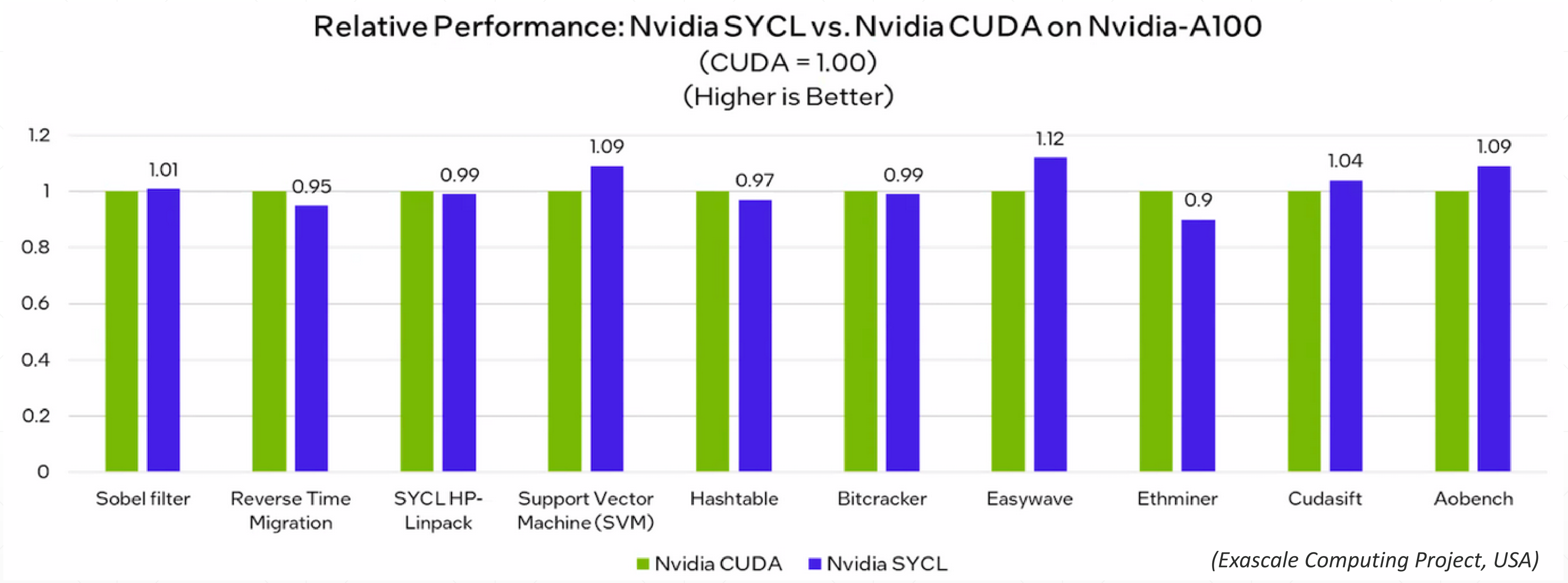
NVIDIA GPU 및 AMD GPU를 위한 컴파일러 플러그인





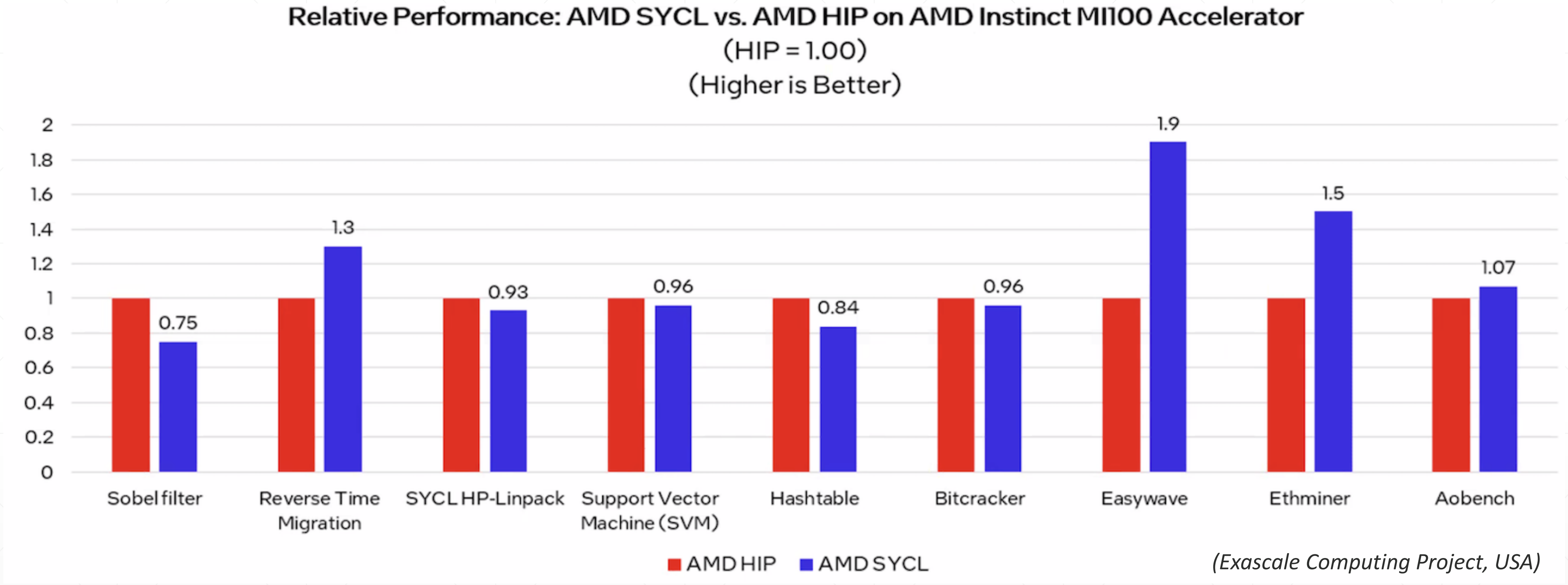
- Intel DPC++ 컴파일러: oneAPI BASE Toolkit
  - OpenCL 백엔드: Intel CPU, GPU(Gen9, 11, Xe) 및 FPGA(Stratix, Aria)에 최적화
  - Level Zero 백엔드: Intel GPU를 위한 low level 오프로드 API
- Codeplay에서 Intel® oneAPI 2023 컴파일러의 최신 바이너리 플러그인을 무료로 다운로드 가능.
  - https://developer.codeplay.com/products/oneapi
  - NVIDIA GPU: A100-PCIe-40GB (sm_80)
  - AMD GPU (베타): AMD Radeon Pro W6600 (gfx1032)

Intel® DPC++ 컴파일러: CUDA 백엔드



- NVIDIA GPU의 네이티브 CUDA*에 필적하는 성능
- 테스트 환경:
  - Intel® Xeon® Platium 8360Y / NVIDIA A100-PCIe-80GB
  - CUDA 11.7

Intel® DPC++ 컴파일러: HIP 백엔드



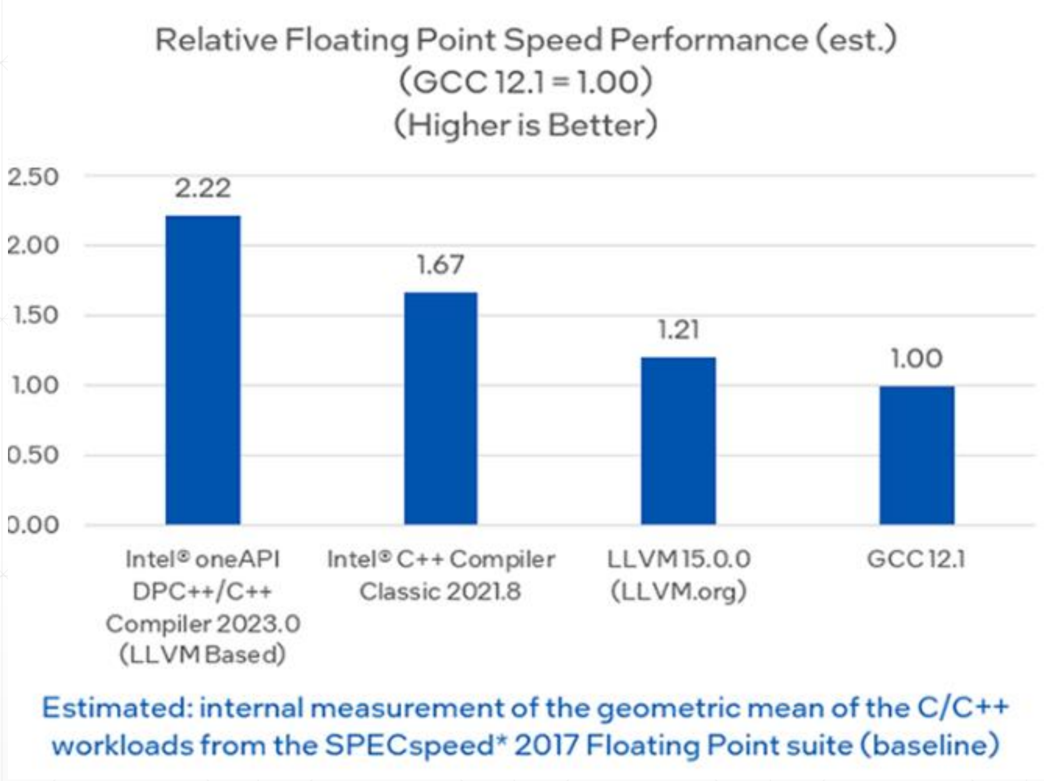
- SYCL 2020 기능의 50% 이상을 구현하다.
- 테스트 환경:
  - Intel® Xeon® Gold 6330 / AMD Instinct MI100
  - RoCm 5.2.1

이기종 프로그래밍 모델 지원

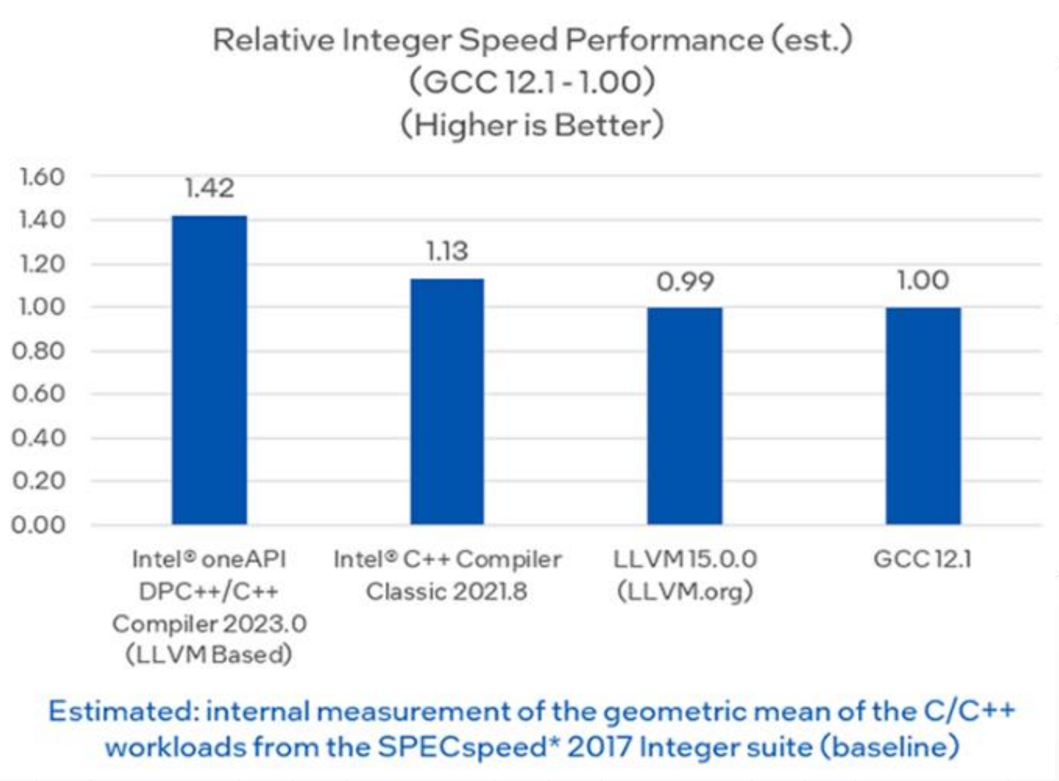
	CUDA	HIP	OpenACC	OpenMP	SYCL/DPC++
Languages	C/C++/Fortran	C/C++/Fortran	C/C++/Fortran	C/C++/Fortran	C++
Abstraction	Low	Low	High	High	Medium
Coding	-	-	Directive-based	Directive-based	C++ lambda
Parallelism	SIMT	SIMT	Fork-join SIMD	Fork-join SIMD	OpenCL
Offload	GPU (NVIDIA)	GPU (NVIDIA/AMD)	GPU (NVIDIA)	CPU/GPU (NVIDIA/AMD/Intel)	CPU/GPU/FPGA (NVIDIA/AMD/Intel)
Compiler	Proprietary	LLVM	PGI/CCE/GCC	PGI/CCE/GCC/LLVM/XL/Intel	LLVM
License	Proprietary	Open-source	Open-source	Open-source	Open-source

- icc/icpc 와 ifort에서는:
  - GPU용 OpenMP*4.0/4.5의 오프로드 기능은 지원되지 않음
  - OpenMP*5.0/5.1/5.2 도 지원 지원되지 않음
- dpcpp/icx/icpx/ifx는 새로운 LLVM 기반 컴파일러이며 Intel® oneAPI Toolkit에 포함
  - GPU용 OpenMP 오프로드 기능은 지원 가능
  - OpenMP*5.0/5.1/5.2도 지원 가능
  - https://www.intel.com/content/www/us/en/developer/articles/technical/openmp-features-and-extensions-supported-in-icx.html

C++ 애플리케이션 성능 향상



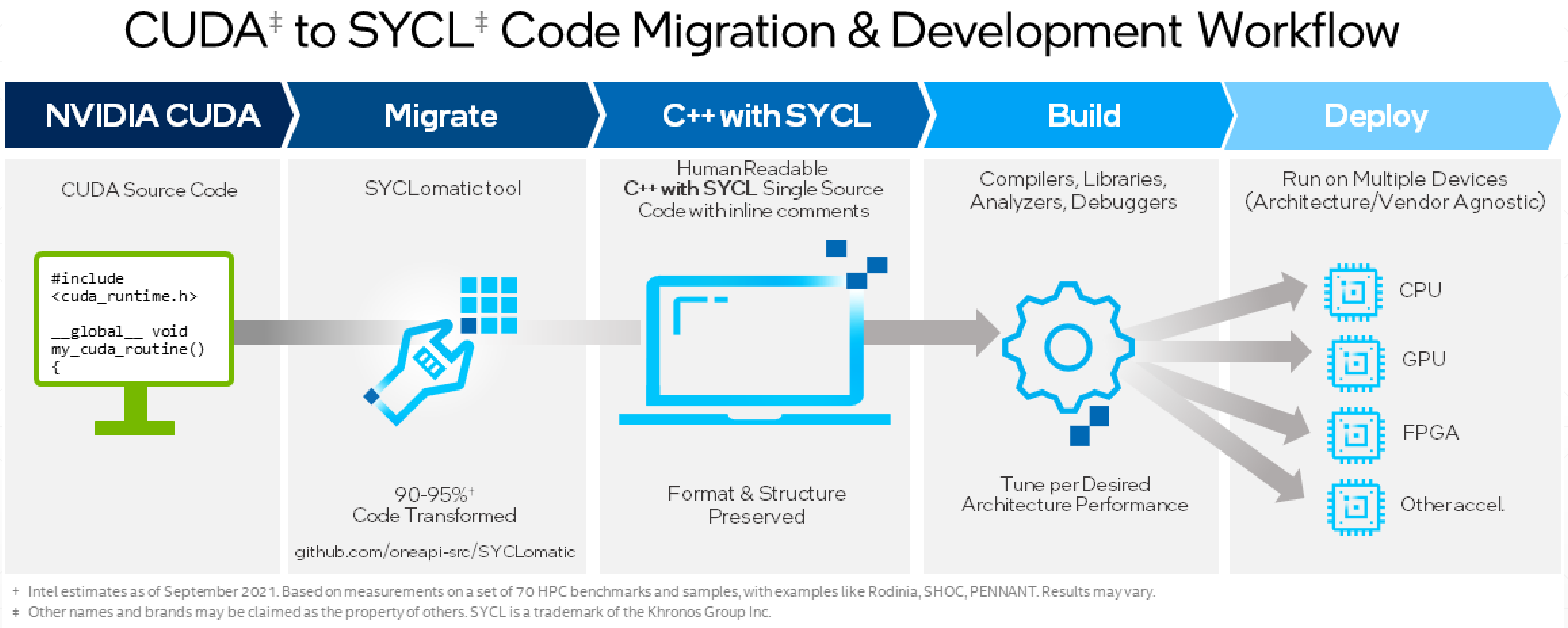
Estimated: internal measurement of the geometric mean of the C/C++ workloads from the SPECspeed* 2017 Floating Point suite (baseline)



Estimated: internal measurement of the geometric mean of the C/C++ workloads from the SPECspeed* 2017 Integer suite (baseline)

- 테스트 환경:
  - Intel® Xeon® Platium 8480
  - SPEC 벤치마크: molecular dynamics, biomedical imaging, ray tracing, fluid dynamics etc

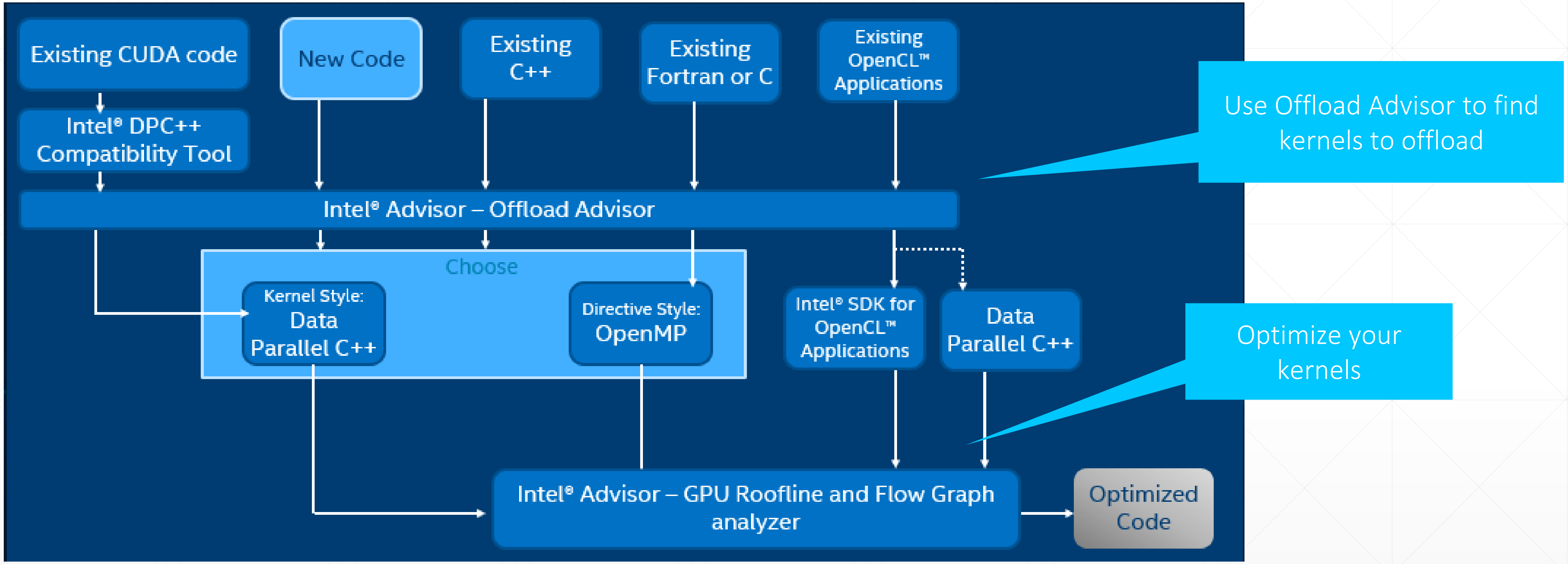
CUDA에서 SYCL로의 손쉬운 전환


CUDA‡ to SYCL‡ Code Migration & Development Workflow
NVIDIA CUDA
Migrate
C++ with SYCL
Build
Deploy
CUDA Source Code
SYCLomatic tool
Human Readable C++ with SYCL Single Source Code with inline comments
Compilers, Libraries, Analyzers, Debuggers
Run on Multiple Devices (Architecture/Vendor Agnostic)
#include <cuda_runtime.h>
__global__ void my_cuda_routine() {
CPU
GPU
FPGA
Other accel.
90-95%† Code Transformed
github.com/oneapi-src/SYCLomatic
Format & Structure Preserved
Tune per Desired Architecture Performance

† Intel estimates as of September 2021. Based on measurements on a set of 70 HPC benchmarks and samples, with examples like Rodinia, SHOC, PENNANT. Results may vary.
‡ Other names and brands may be claimed as the property of others. SYCL is a trademark of the Khronos Group Inc.

- 이미 CUDA*로 작성된 코드를 DPCPP로 전환하고자 하는 개발자를 지원하며, 사람이 해독할 수 있는 코드를 생성
  - (일반적으로 코드의 90~95%를 자동으로 마이그레이션 가능)
- 개발자가 애플리케이션 마이그레이션을 완료할 수있도록 인라인 코멘트를 제공

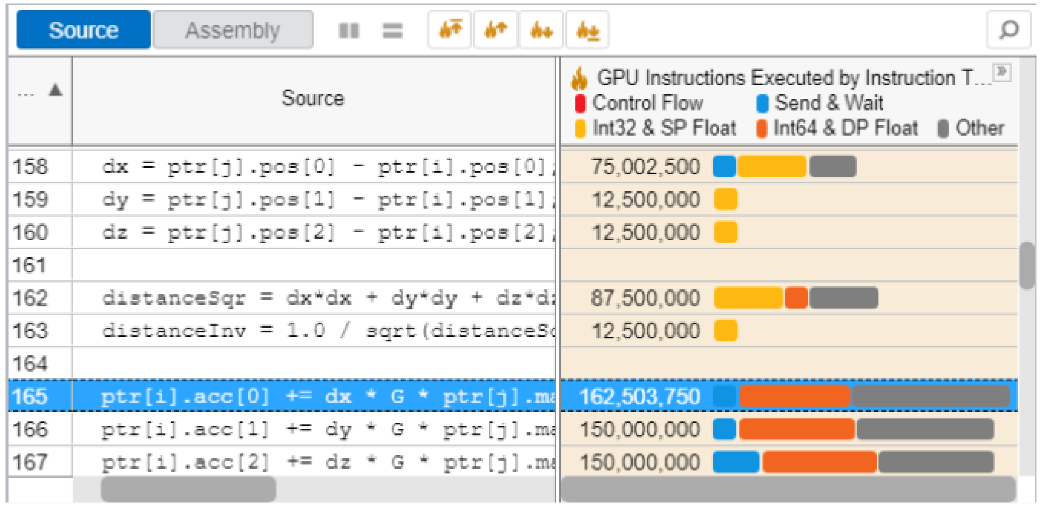
최신 아키텍처를 위한 고성능 코드의 설계와 최적화


Existing CUDA code
New Code
Existing C++
Existing Fortran or C
Existing OpenCL™ Applications
Intel® DPC++ Compatibility Tool
Intel® Advisor – Offload Advisor
Use Offload Advisor to find kernels to offload
Choose
Kernel Style: Data Parallel C++
Directive Style: OpenMP
Intel® SDK for OpenCL™ Applications
Data Parallel C++
Optimize your kernels
Intel® Advisor – GPU Roofline and Flow Graph analyzer
Optimized Code

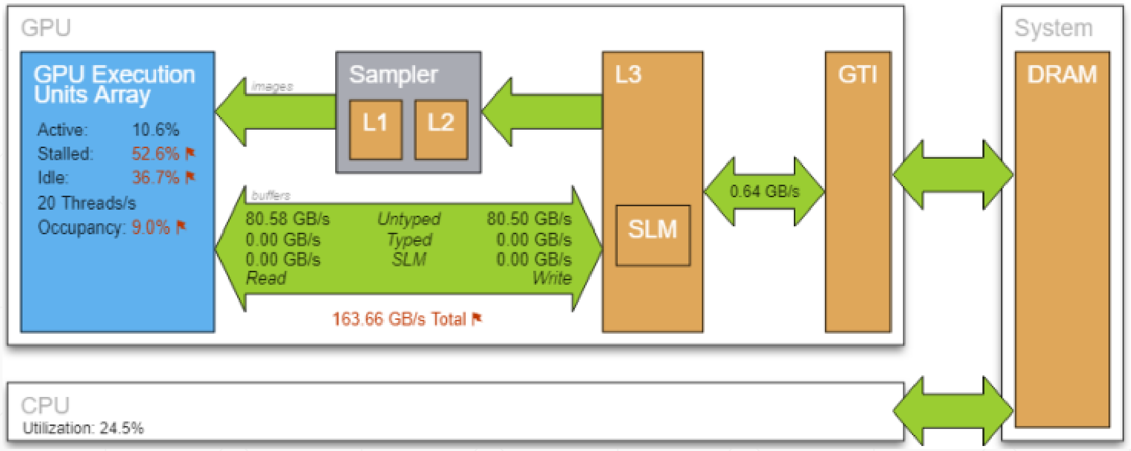
- 새로운 Intel® 오프로드 어드바이저
  - GPU로 오프로드할 최적의 코드 영역을 찾아 GPU 결과를 추측하고 성능 병목 현상을 파악
  - GPU로 이행된 코드의 병목 현상을 식별하여 성능이 시스템의 최대값에 얼마나 근접했는지 확인 가능

Intel® VTune™ 프로파일러

- SYCL 코드 분석
  - 가장 많은 시간을 소비하는 SYCL 코드 확인
- Intel의 CPU, GPU, FPGA를 위한 튜닝
  - 지원되는 하드웨어 가속기에 최적화
- 오프로드 최적화
  - OpenMP* 오프로드 성능 튜닝
- 광범위한 성능 프로파일
  - CPU, GPU, FPGA, 스레드, 메모리, 캐시
- 일반 언어 지원
  - SYCL, C, C++, Fortran, Python, Go, Java, 또는 이들의 조합

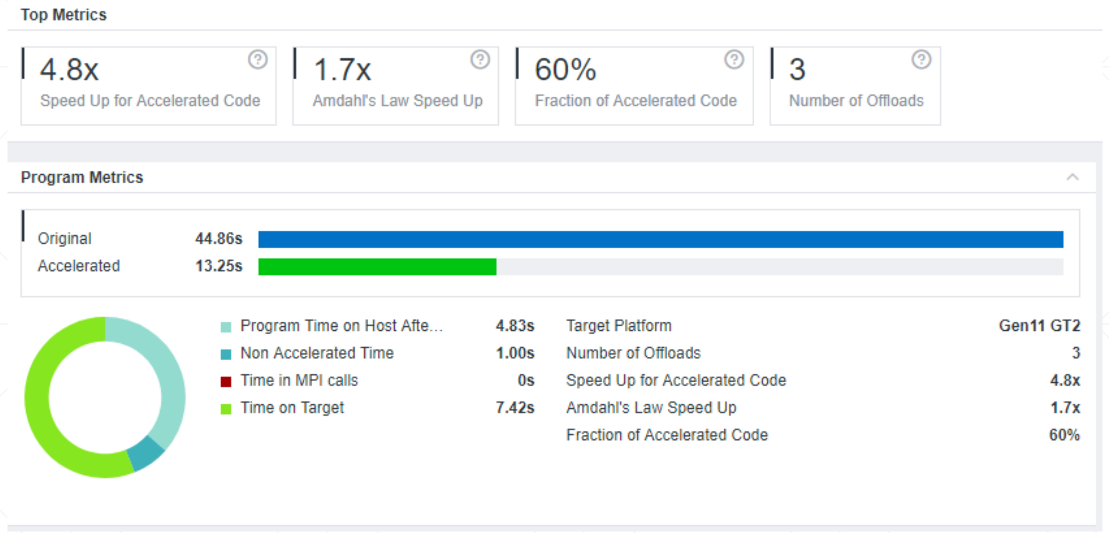


... ▲	Source	GPU Instructions Executed by Instruction T... 🟥 Control Flow 🟦 Send & Wait 🟨 Int32 & SP Float 🟧 Int64 & DP Float ⬛ Other
158	`dx = ptr[j].pos[0] - ptr[i].pos[0];`	75,002,500
159	`dy = ptr[j].pos[1] - ptr[i].pos[1];`	12,500,000
160	`dz = ptr[j].pos[2] - ptr[i].pos[2];`	12,500,000
161		
162	`distanceSqr = dx*dx + dy*dy + dz*dz`	87,500,000
163	`distanceInv = 1.0 / sqrt(distanceSq`	12,500,000
164		
165	`ptr[i].acc[0] += dx * G * ptr[j].ma`	162,503,750
166	`ptr[i].acc[1] += dy * G * ptr[j].ma`	150,000,000
167	`ptr[i].acc[2] += dz * G * ptr[j].ma`	150,000,000



Intel® 어드바이저

- 오프로드 모델링
  - 액셀러레이터에 오프로드된 성능을 추정
- 루프 라인 분석
  - CPU/GPU 코드의 계산 및 메모리 최적화
- 벡터화 어드바이저
  - 벡터화와 최적화
- 스레딩 어드바이저
  - 스레드화되지 않은 애플리케이션에 효율적인 스레드화
- 플로우 그래프 분석기
  - 효율적인 흐름그래프 생성 및 분석



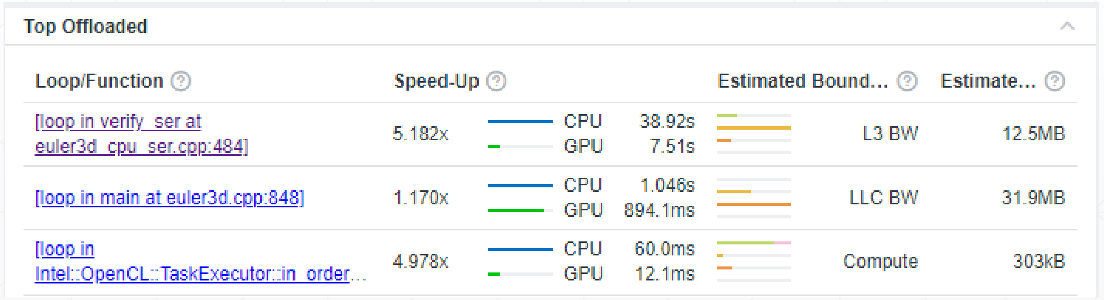
Top Metrics

4.8x	1.7x	60%	3
Speed Up for Accelerated Code	Amdahl's Law Speed Up	Fraction of Accelerated Code	Number of Offloads

Program Metrics

Original	44.86s
Accelerated	13.25s

Program Time on Host Afte...	4.83s	Target Platform	Gen11 GT2
Non Accelerated Time	1.00s	Number of Offloads	3
Time in MPI calls	0s	Speed Up for Accelerated Code	4.8x
Time on Target	7.42s	Amdahl's Law Speed Up	1.7x
		Fraction of Accelerated Code	60%



Top Offloaded

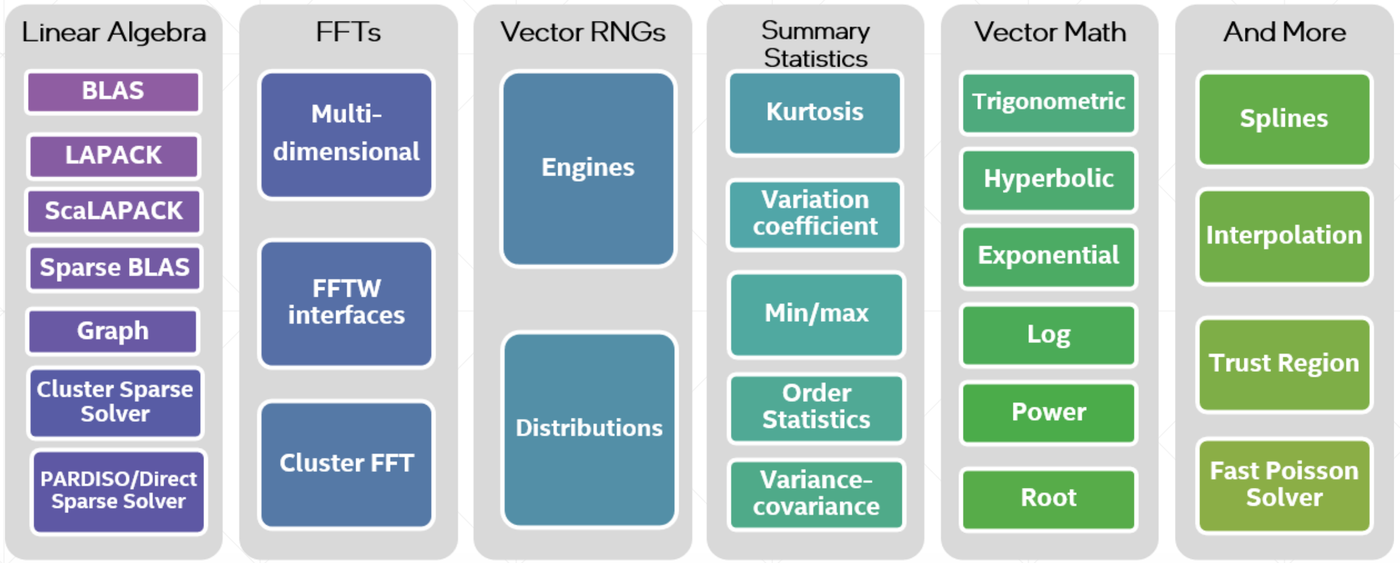
Loop/Function	Speed-Up			Estimated Bound...	Estimate...
[loop in verify_ser at euler3d_cpu_ser.cpp:484]	5.182x	CPU GPU	38.92s 7.51s	L3 BW	12.5MB
[loop in main at euler3d.cpp:848]	1.170x	CPU GPU	1.046s 894.1ms	LLC BW	31.9MB
[loop in Intel::OpenCL::TaskExecutor::in_order...	4.978x	CPU GPU	60.0ms 12.1ms	Compute	303kB

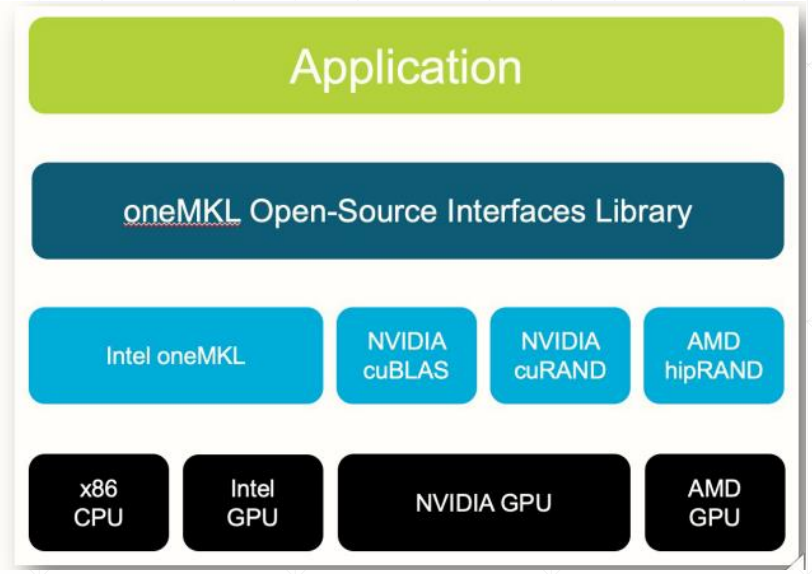
Intel® 성능 라이브러리

- 풍부한 기능
  - 모든 사용 사례에 최적화된 성능 라이브러리를 제공
  - 스레드, 오프로드, 수학, 데이터 분석, 데이터 처리 렌더링
- 하드웨어의 가치를 극대화
  - HPC 도메인 별 기능의 속도를 높이도록 설계
- 선택의 자유
  - 최고 성능을 위해 각 플랫폼에 맞게 최적화

oneAPI Math Kernel Library **oneMKL**	oneAPI Deep Neural Network Library **oneDNN**
oneAPI Video Processing Library **oneVPL**	oneAPI Data Analytics Library **oneDAL**
oneAPI Threading Building Blocks **oneTBB**	oneAPI Collective Communications Library **oneCCL**
oneAPI DPC++ Library **oneDPL**	Ray Tracing
oneAPI Image Processing Library* **oneIPL**	oneAPI Data Transformation Library* oneDTL

Intel® oneMKL: HPC 성능을 향상시키는 수치 연산 라이브러리





- CPU와 GPU 아키텍처용으로 최적화된 인터페이스가 주요 계산 기능에 추가
  - BLAS와 LAPACK 루틴
  - Sparse BLAS 루틴
  - 난수 제너레이터(RNG)
  - 최적화된 벡터 수학(VM) 루틴
  - 고속 푸리에 변환(FFT)

Intel® oneMKL: SYCL

DPC++ API

```
using namespace cl::sycl;

constexpr size_t N = 256;
float A[N] = {0};


buffer<float, 1> A_buf{A, A+N};
buffer<float, 1> R_buf{N};


device dev{gpu_selector()};
queue q{dev};


oneapi::mkl::vm::sin(q, N,
     A_buf, R_buf,
     oneapi::mkl::vm::mode::la);
```

host memory

create DPC++ buffers

choose device and queue

calculate R = sin(A)

C API

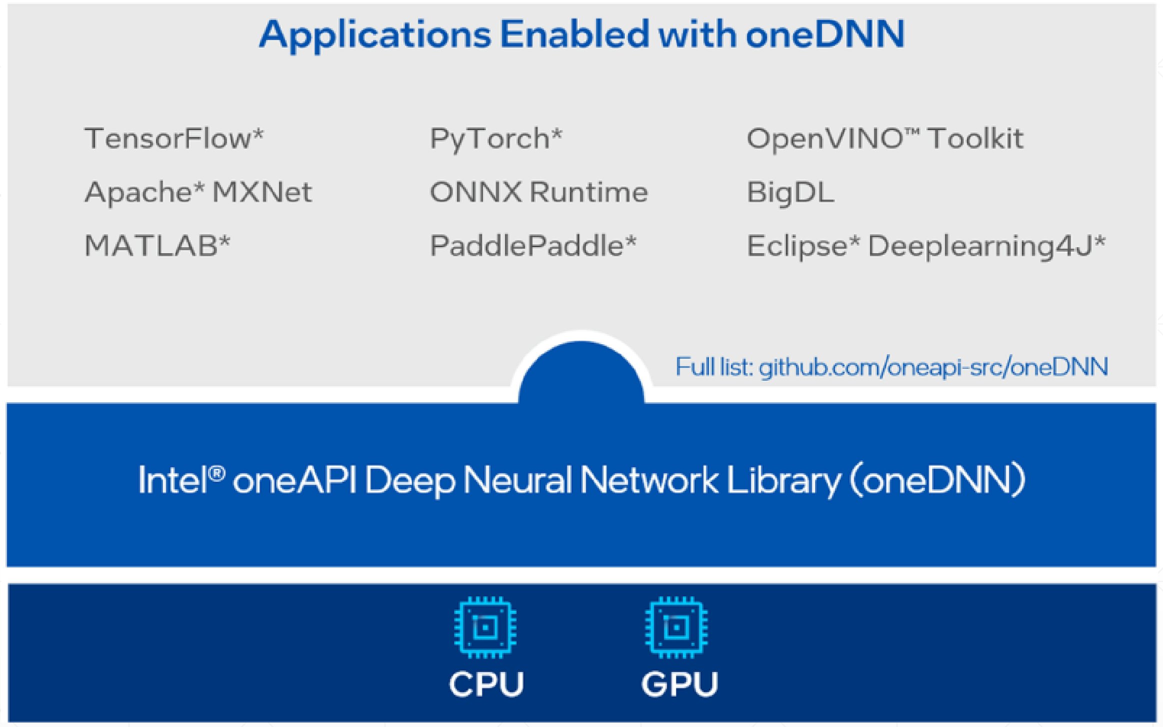
```
constexpr size_t N = 256;
float A[N] = {0}, R[N];


vmsSin(N, A, R, VML_LA);
```

Intel® oneMKL: OpenMP 오프로드

```
double *A = ..., *B = ..., *C = ...;
const int dnum = 0;

#pragma omp target map(C[0:sizeC])
{
   Initialize C
}

#pragma omp target data map(to:A[…], B[…]) map(tofrom:C[…]) device(dnum)
{
   #pragma omp target variant dispatch device(dnum) use_device_ptr(A, B, C)
   {
      cblas_dgemm(CblasColMajor, CblasNoTrans, CblasNoTrans, m, n, k, alpha, A, lda,
B, ldb, beta, C, ldc);
   }

}
```

Intel® oneDNN: Deep Neural Network 라이브러리

- 고성능 딥러닝 프레임워크를 만들 수 있도록 지원
- 명령 세트와 성능 최적화의 복잡성를 추상화
- Intel의 CPU와 GPU에서 동일한 API를 제공
- 오픈소스를 통한 커뮤니티 기여


Applications Enabled with oneDNN
TensorFlow*
PyTorch*
OpenVINO™ Toolkit
Apache* MXNet
ONNX Runtime
BigDL
MATLAB*
PaddlePaddle*
Eclipse* Deeplearning4J*
Full list: github.com/oneapi-src/oneDNN
Intel® oneAPI Deep Neural Network Library (oneDNN)
CPU
GPU

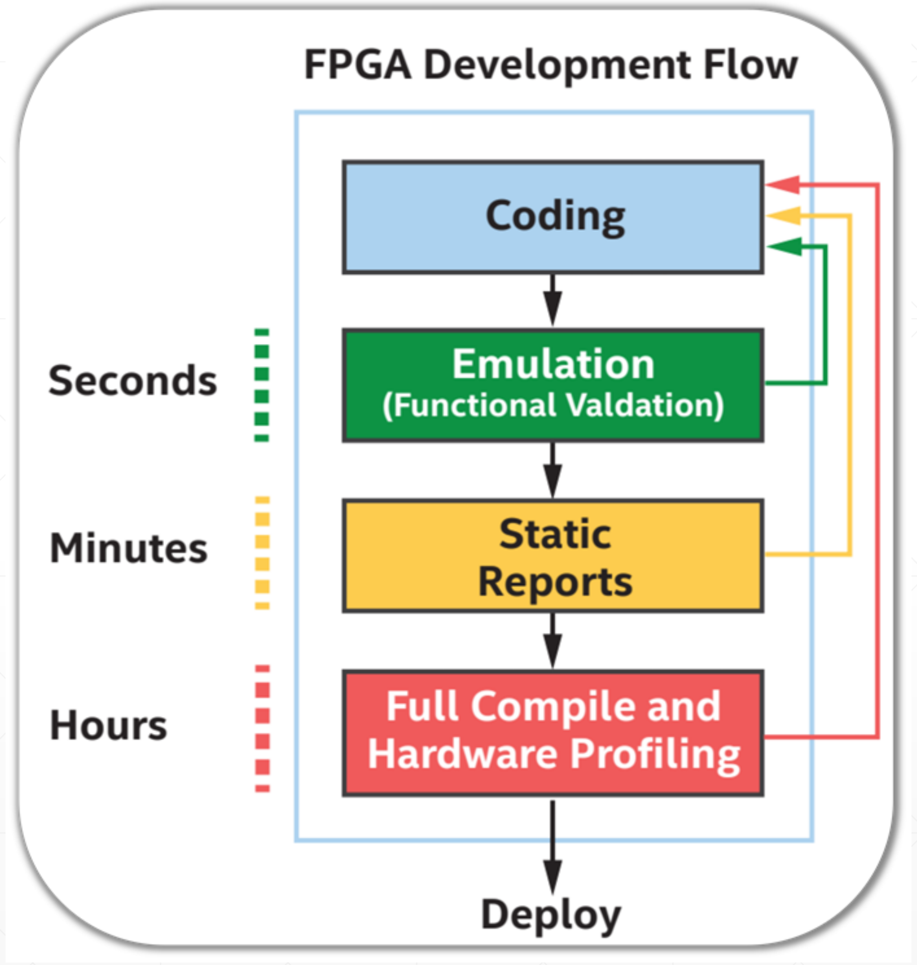
```
>>> import tensorflow as tf
>>> tf.config.list_physical_devices()
[PhysicalDevice(name='/physical_device:CPU:0', device_type='CPU’),
 PhysicalDevice(name='/physical_device:GPU:0', device_type=‘GPU’),
 PhysicalDevice(name='/physical_device:XPU:0', device_type=‘XPU')]

>>> a = tf.random.normal(shape=[5], dtype=tf.float32)  # Runs on CPU
>>> b = tf.nn.relu(a)                                   # Runs on NVIDIA GPU

>>> with tf.device("/XPU:0"):                           # Device selection
...   c = tf.nn.relu(a)                                 # Runs on Intel XPU
```

Intel® oneDDN: API 비교

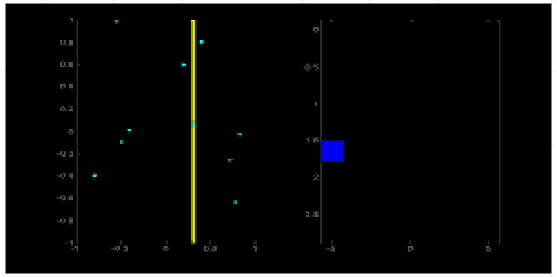
src/dst tensor	
cudnnCreateTensorDescriptor()	memory::desc()
cudnnSetTensor4dDescriptor()	sycl_interop::make_memory()
↓	
filter tensor	
cudnnCreateFilterDescriptor()	memory::desc()
cudnnSetFilter4dDescriptor()	sycl_interop::make_memory()
↓	
convolution descriptor	
cudnnCreateConvolutionDescriptor()	convolution_forward::desc()
cudnnSetConvolution2dDescriptor()	convolution_forward::primitive_desc()
↓	
convolution algorithm	
cudnnGetConvolutionForwardAlgorithm()	convolution_forward()
cudnnGetConvolutionForwardWorkspaceSize()	sycl_interop::make_memory()
↓	
convolution	
cudnnConvolutionForward()	execute()

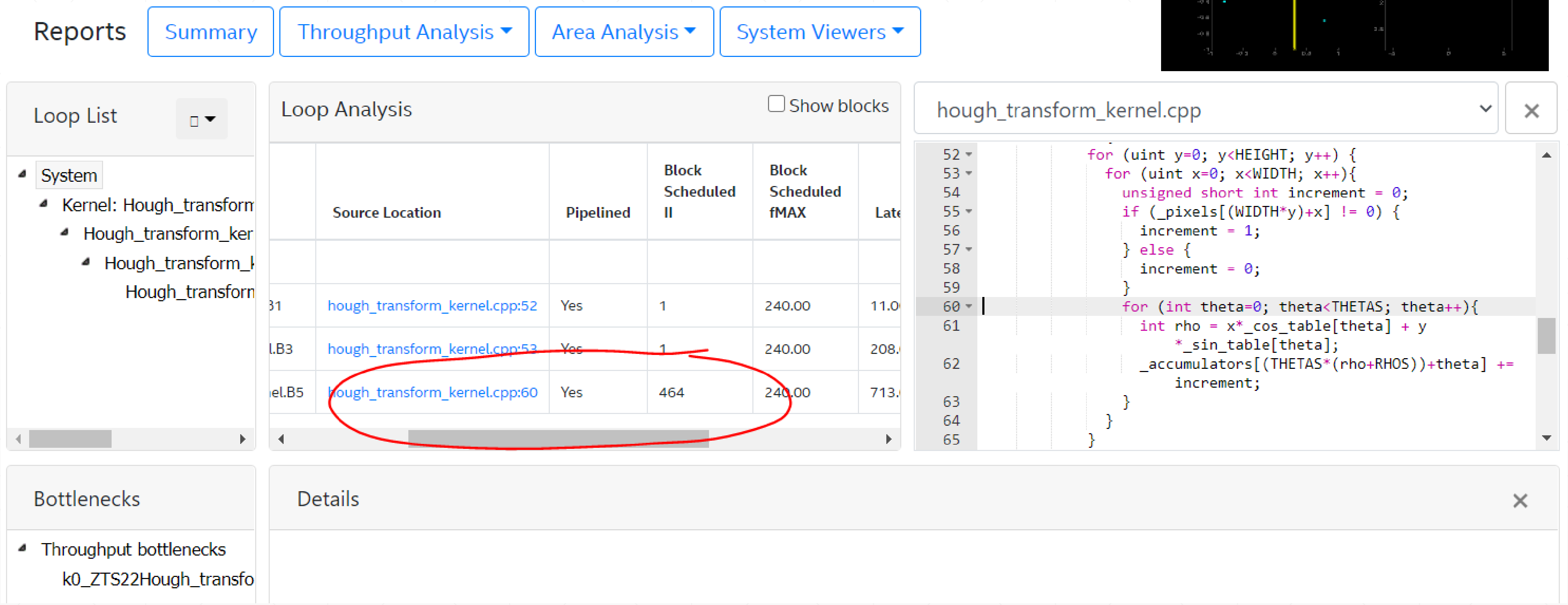
Intel® oneAPI에 대한 FPGA 개발 흐름

- FPGA의 가장 큰 문제 중 하나는 컴파일 시간
- FPGA 개발 흐름은 컴파일 실행을 최소화하도록 조정
- 기능 유효성 검사
  - CPU에서 컴파일되고 실행되는 개발 플랫폼의 에뮬레이션 사용
  - 버그가 발견되어 수정 필요 시 훨씬 빠른 처리 시간 가능
- 정적 성능 분석
  - 메모리, 성능, 데이터 흐름 병목 현상을 식별하는 정보
  - 병목 현상을 해결하기 위한 최적화 기술에 대한 제안
- 전체 컴파일
  - Vtune 성능 분석기에서 데이터 패턴 종속 병목 현상을 식별



FPGA 개발 흐름: 정적 성능 분석





Reports | Summary | Throughput Analysis ▾ | Area Analysis ▾ | System Viewers ▾

Loop List

- System
  - Kernel: Hough_transform
    - Hough_transform_ker
      - Hough_transform_l
        - Hough_transform

Loop Analysis ☐ Show blocks

	Source Location	Pipelined	Block Scheduled II	Block Scheduled fMAX	Late
31	hough_transform_kernel.cpp:52	Yes	1	240.00	11.0
l.B3	hough_transform_kernel.cpp:53	Yes	1	240.00	208.
el.B5	hough_transform_kernel.cpp:60	Yes	464	240.00	713.

hough_transform_kernel.cpp

```
52        for (uint y=0; y<HEIGHT; y++) {
53          for (uint x=0; x<WIDTH; x++){
54            unsigned short int increment = 0;
55            if (_pixels[(WIDTH*y)+x] != 0) {
56              increment = 1;
57            } else {
58              increment = 0;
59            }
60            for (int theta=0; theta<THETAS; theta++){
61              int rho = x*_cos_table[theta] + y
                    *_sin_table[theta];
62              _accumulators[(THETAS*(rho+RHOS))+theta] +=
                    increment;
63            }
64          }
65        }
```

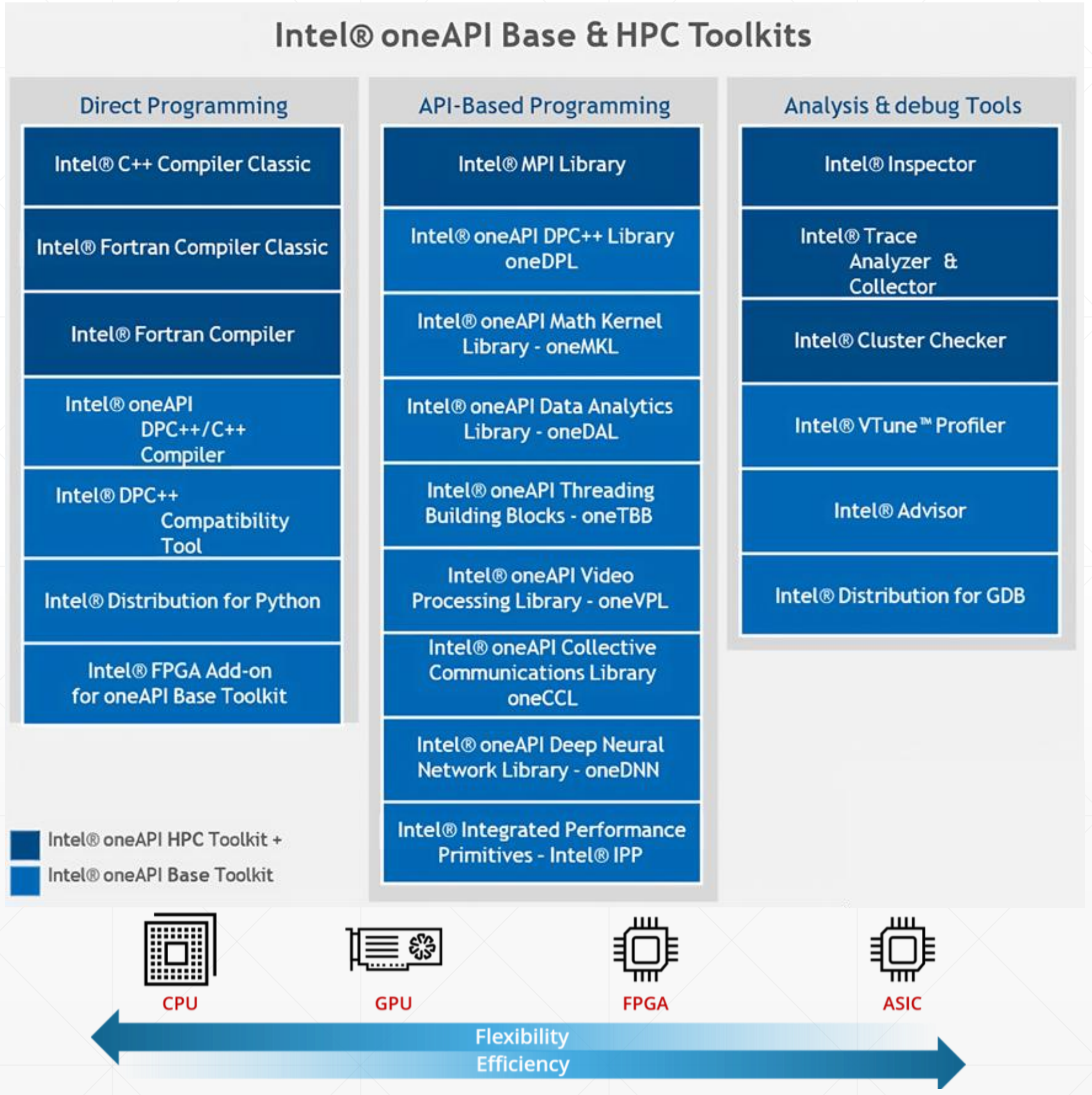
Bottlenecks

- Throughput bottlenecks
  - k0_ZTS22Hough_transfo

Details

Intel® oneAPI HPC 툴킷: 확장 가능하고 빠른 애플리케이션 구현

- Intel® oneAPI BASE Toolkit에 추가 가능
- C++, Fortran, SYCL*, OpenMP* 및 MPI로 엔터프라이즈, 클라우드, HPC 및 AI를 위한 고성능의 확장 가능한 병렬 애플리케이션을 개발 가능
- 적은 노력으로 빠르고 안정적이며 확장 가능한 산업 표준병렬 코드 생성
- Intel® 제온® 프로세서, Intel® 코어™ 프로세서 및 Intel 가속기에서의 성능 향상


Intel® oneAPI Base & HPC Toolkits
Direct Programming
Intel® C++ Compiler Classic
Intel® Fortran Compiler Classic
Intel® Fortran Compiler
Intel® oneAPI DPC++/C++ Compiler
Intel® DPC++ Compatibility Tool
Intel® Distribution for Python
Intel® FPGA Add-on for oneAPI Base Toolkit
API-Based Programming
Intel® MPI Library
Intel® oneAPI DPC++ Library oneDPL
Intel® oneAPI Math Kernel Library - oneMKL
Intel® oneAPI Data Analytics Library - oneDAL
Intel® oneAPI Threading Building Blocks - oneTBB
Intel® oneAPI Video Processing Library - oneVPL
Intel® oneAPI Collective Communications Library oneCCL
Intel® oneAPI Deep Neural Network Library - oneDNN
Intel® Integrated Performance Primitives - Intel® IPP
Analysis & debug Tools
Intel® Inspector
Intel® Trace Analyzer & Collector
Intel® Cluster Checker
Intel® VTune™ Profiler
Intel® Advisor
Intel® Distribution for GDB
Intel® oneAPI HPC Toolkit +
Intel® oneAPI Base Toolkit
CPU
GPU
FPGA
ASIC
Flexibility
Efficiency

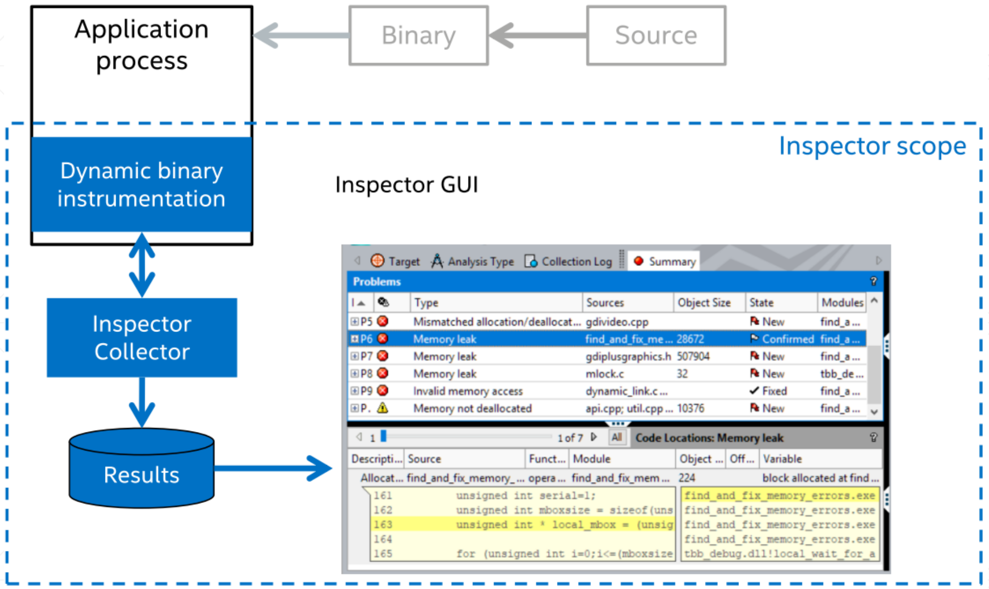
Intel® Inspector: 더 적은 비용으로 신뢰성 높은 애플리케이션 개발

- C, C++ 및 Fortran 애플리케이션의 메모리 에러와 스레드화 에러를 검출하는데 용이

- 별도의 컴파일러나 빌드가 필요 없음
  - 릴리스 빌드 바이너리를 사용
  - GUI는 Windows와 Linux OS 상에서 이용 가능
  - Microsoft* Visual Studio*로 통합 가능

- 메모리 오류
  - 메모리 누설
  - 할당과 해방의 API 미스매치
  - 일관성 없는 메모리 API 사용
  - 부정한 메모리 액세스

- 스레드화 오류
  - 데이터 경합
  - Deadlock



I		Type	Sources	Object Size	State	Modules
P5		Mismatched allocation/deallocat...	gdivideo.cpp		New	find_a...
P6		Memory leak	find_and_fix_me...	28672	Confirmed	find_a...
P7		Memory leak	gdiplusgraphics.h	507904	New	find_a...
P8		Memory leak	mlock.c	32	New	tbb_de...
P9		Invalid memory access	dynamic_link.c...		✓ Fixed	find_a...
P.		Memory not deallocated	api.cpp; util.cpp...	10376	New	find_a...

Descripti...	Source	Funct...	Module	Object ...	Off...	Variable
Allocat...	find_and_fix_memory_...	opera...	find_and_fix_mem...	224		block allocated at find...

```
161         unsigned int serial=1;
162         unsigned int mboxsize = sizeof(uns
163         unsigned int * local_mbox = (unsig
164
165         for (unsigned int i=0;i<=(mboxsize
```

```
find_and_fix_memory_errors.exe
find_and_fix_memory_errors.exe
find_and_fix_memory_errors.exe
find_and_fix_memory_errors.exe
tbb_debug.dll!local_wait_for_a
```

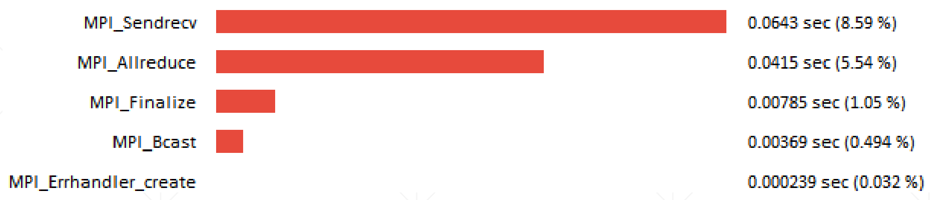
Features	Memory Analysis	Threading Analysis	Persistence Memory
Collapse multiple “sightings” to one error	✓	✓	✓
Suppress, Filter, Workflow Management	✓	✓	✓
Visual Studio Integration (Windows)	✓	✓	✓
Command line for automated tests	✓	✓	✓
Timeline visualization	✓	✓	
Memory growth during a transaction	✓		
Trigger debugger breakpoints	✓	✓	

Intel® Trace Analyzer: 소프트웨어의 병목 현상을 빠르게 파악

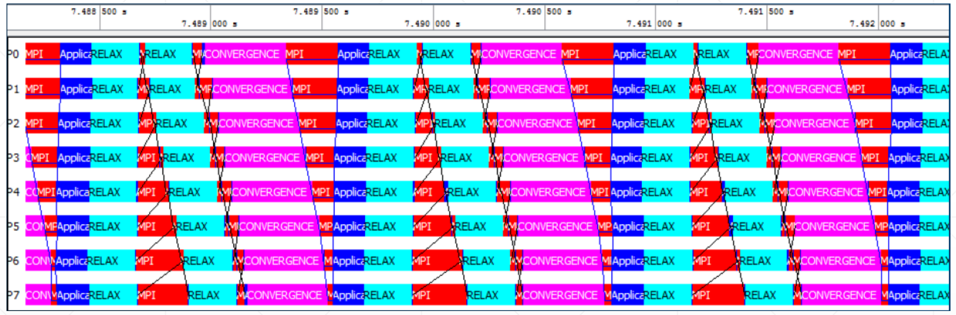
- 지원되는 디버그 도구
  - GDB
  - Allinea DDT
- MPI 애플리케이션을 위한 성능 분석 도구
  - 통신 hotspot 표시
  - 병렬 프로그램의 동작을 시각화
  - 일반 MPI 오류를 분석
- 새로운 경량 프로파일로 100K+ MPI 랭크를 프로파일 가능

Top MPI functions

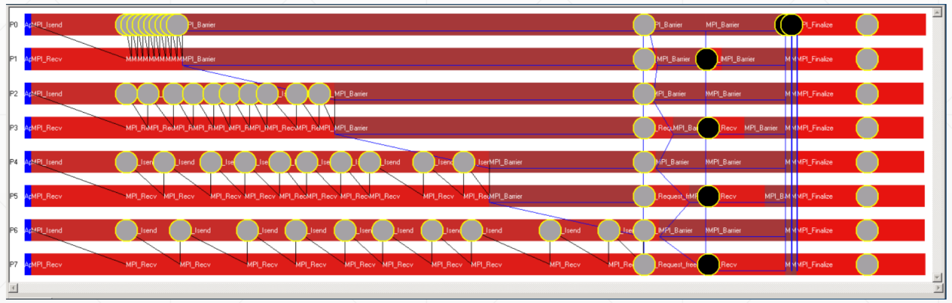
This section lists the most active MPI functions from all MPI calls in the application.



MPI event timeline

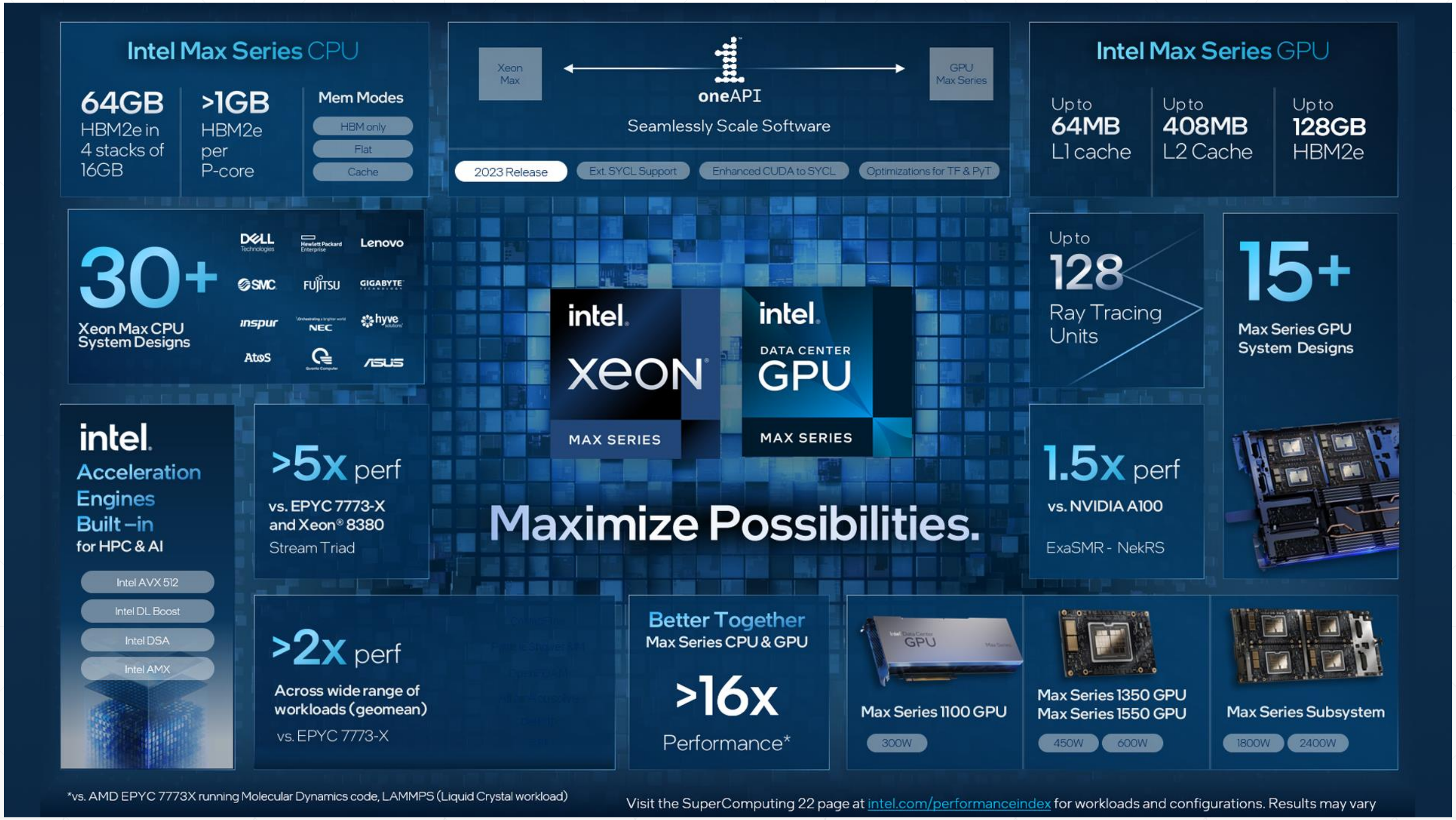


MPI correctness checking



● warnings ● errors

Intel® 맥스 시리즈



Intel Max Series CPU

64GB HBM2e in 4 stacks of 16GB

>1GB HBM2e per P-core

Mem Modes
- HBM only
- Flat
- Cache

Xeon Max ⟷ oneAPI ⟷ GPU Max Series

oneAPI
Seamlessly Scale Software

2023 Release | Ext. SYCL Support | Enhanced CUDA to SYCL | Optimizations for TF & PyT

Intel Max Series GPU

Up to 64MB L1 cache

Up to 408MB L2 Cache

Up to 128GB HBM2e

30+ Xeon Max CPU System Designs

DELL Technologies
Hewlett Packard Enterprise
Lenovo
SMC
FUJITSU
GIGABYTE
inspur
NEC
hyve
Atos
ASUS

intel XEON MAX SERIES

intel DATA CENTER GPU MAX SERIES

Up to 128 Ray Tracing Units

15+ Max Series GPU System Designs

intel Acceleration Engines Built–in for HPC & AI
- Intel AVX 512
- Intel DL Boost
- Intel DSA
- Intel AMX

>5x perf
vs. EPYC 7773-X and Xeon® 8380
Stream Triad

Maximize Possibilities.

1.5x perf
vs. NVIDIA A100
ExaSMR - NekRS

>2x perf
Across wide range of workloads (geomean)
vs. EPYC 7773-X

Better Together
Max Series CPU & GPU
>16x
Performance*

Max Series 1100 GPU
300W

Max Series 1350 GPU
Max Series 1550 GPU
450W 600W

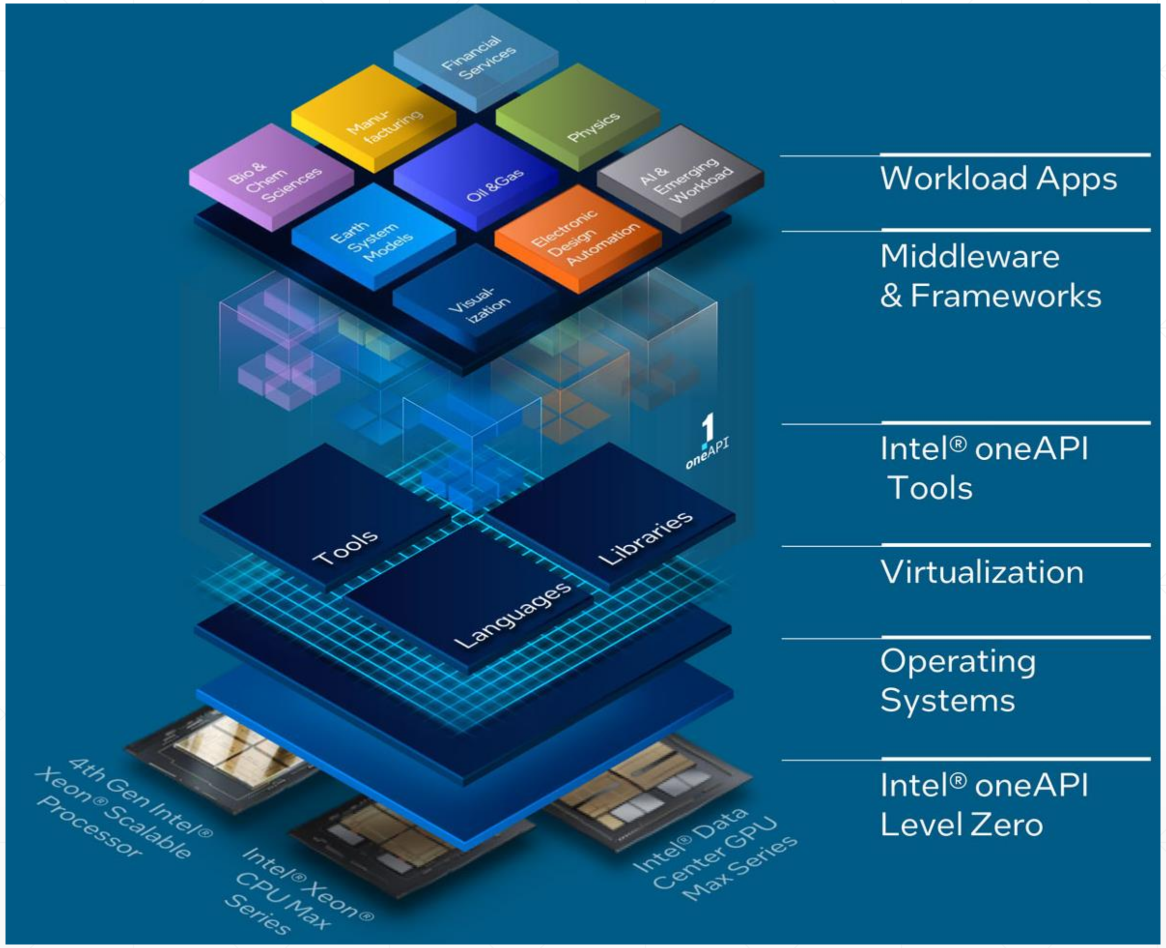
Max Series Subsystem
1800W 2400W

*vs. AMD EPYC 7773X running Molecular Dynamics code, LAMMPS (Liquid Crystal workload)

Visit the SuperComputing 22 page at intel.com/performanceindex for workloads and configurations. Results may vary

Intel® 맥스 시리즈

- Intel® 제온® CPU 맥스
  - 연결된 4개 타일로 구성된 최대 56개 퍼포먼스 코어
  - 고대역폭 메모리(HBM) 구성에서 실행 가능
  - 350W
- Intel® 맥스 시리즈 GPU
  - Intel® 맥스 시리즈 1100 GPU:
    - 56개의 X^e^ 코어 / 64GB HBM / 300W
    - 인텔 X^e^ 링크 브릿지를 통해 여러 카드 연결 가능
  - Intel® 맥스 시리즈 1350 GPU
    - 112개 X^e^ 코어 / 96GB HBM / 450W
  - Intel® 맥스 시리즈 1550 GPU
    - 128개 X^e^ 코어 / 128GB HBM / 600W
- 개방형 표준 기반 교차 아키텍처 프로그래밍 프레임워크인 oneAPI로 통합이 가능



Intel® oneAPI 2023 새로운 기능

- 컴파일러 및 SYCL 지원
  - Intel® oneAPI DPC++/C++ 컴파일러는 BF16을 완벽하게 지원하여 AI 가속을 제공
  - NVIDA 및 AMD GPU 이용하여 CodePlay의 oneAPI 플러그인을 지원
  - SYCLomatic 통해 cuBLAS 및 cuDNN과 같은 CUDA* 라이브러리 호출을 SYCL 및 oneAPI 라이브러리에 변환 가능

- 고성능 컴퓨팅을 위한 최신 Fortran 컴파일러
  - Fortran 2003, Fortran 2008 및 Fortran 2018의 모든 기능과 더 많은 OpenMP 5.x 기능을 지원
  - CPU에서 Co-Array를 지원하며 GPU에서 DO CONCURRENT를 지원

- 성능 라이브러리
  - Intel® oneAPI Math Kernel Library는 데이터 센터 GPU 성능 최적화 포함
  - 새로운 FFT, 1D 및 2D 최적화, 난수 생성기, Sparse BLAS 및 LAPACK 제공
  - Intel ® MPI Library는 GPU 버퍼를 사용하여 Collective 기능의 성능 향상 가능

- 분석 및 디버그
  - Intel® VTune™
    - Intel® Xeon® Max에서 HBM 이용하여 성능 향상 가능
    - Intel® 데이터 센터 Max에서 X^e Link 관련한 CPU/GPU 불균형 및 대역폭 병목 현상 문제 표시

Intel® 맥스 시리즈 성능

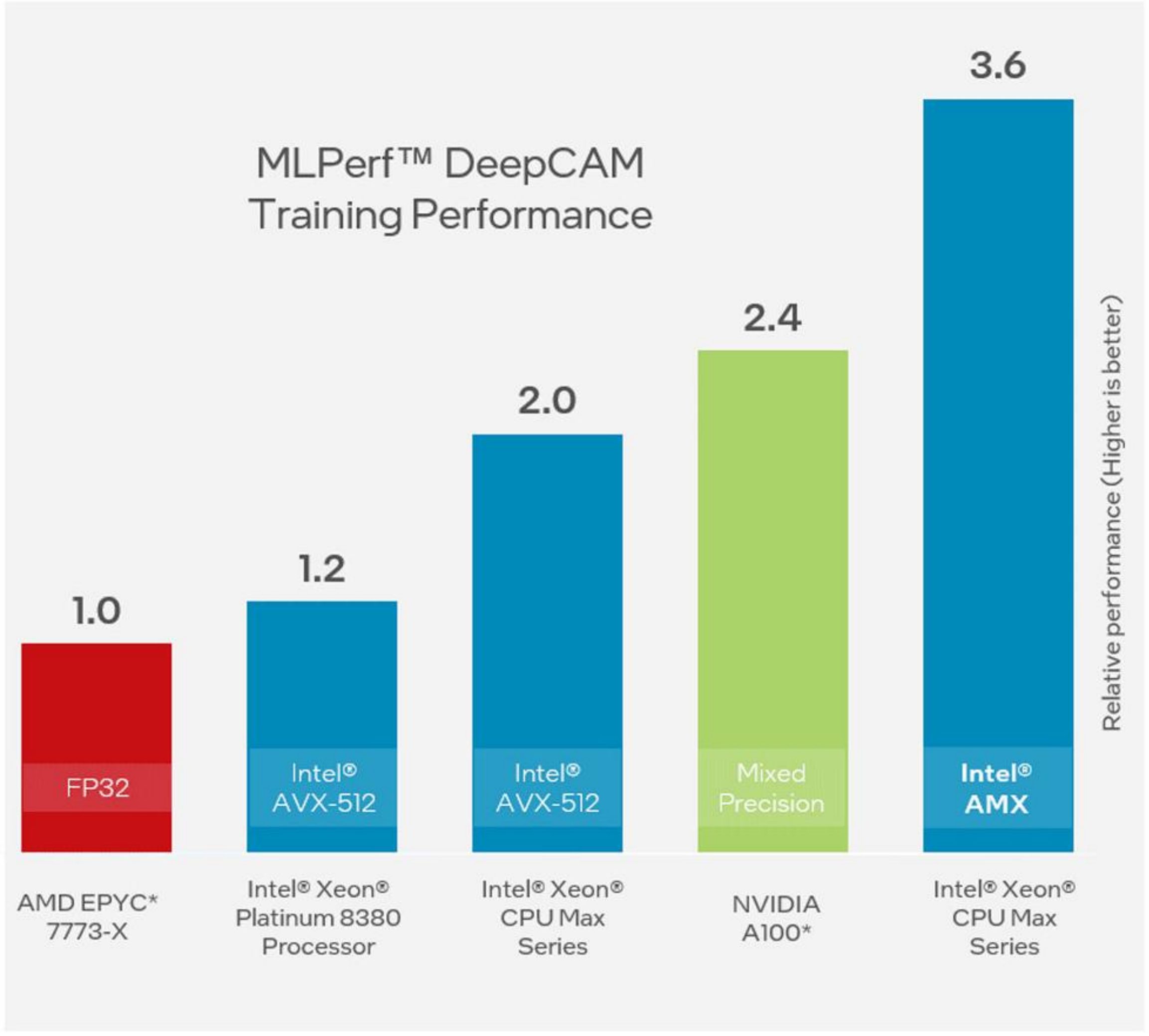
intel.

Advanced Matrix Extensions

intel. xeon MAX SERIES

Performance Leap in Deep Learning Inference & Training

- INT8 (all sign combinations) with INT32 Accumulation
- Bfloat16 with IEEE SP Accumulation
- Full Intel® Architecture (IA) Programmability
- Very Low Latency



Intel® 맥스 시리즈 성능

Life & Material Science

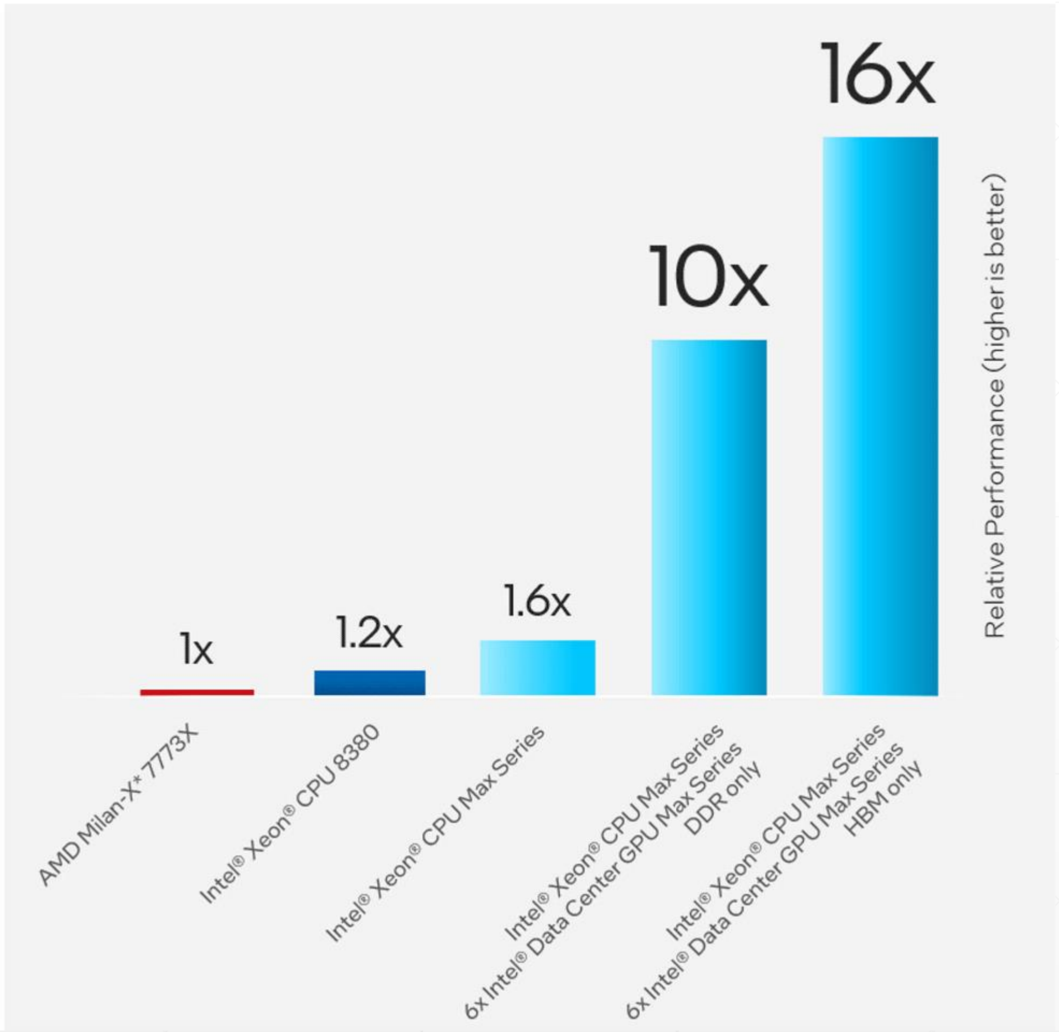
Speeding Exascale Material Discovery with Compute Density and HBM for LAMMPS (Liquid Crystal)

Performance results are **based on testing by Intel as of October 28, 2022** and may not reflect all publicly available security updates.

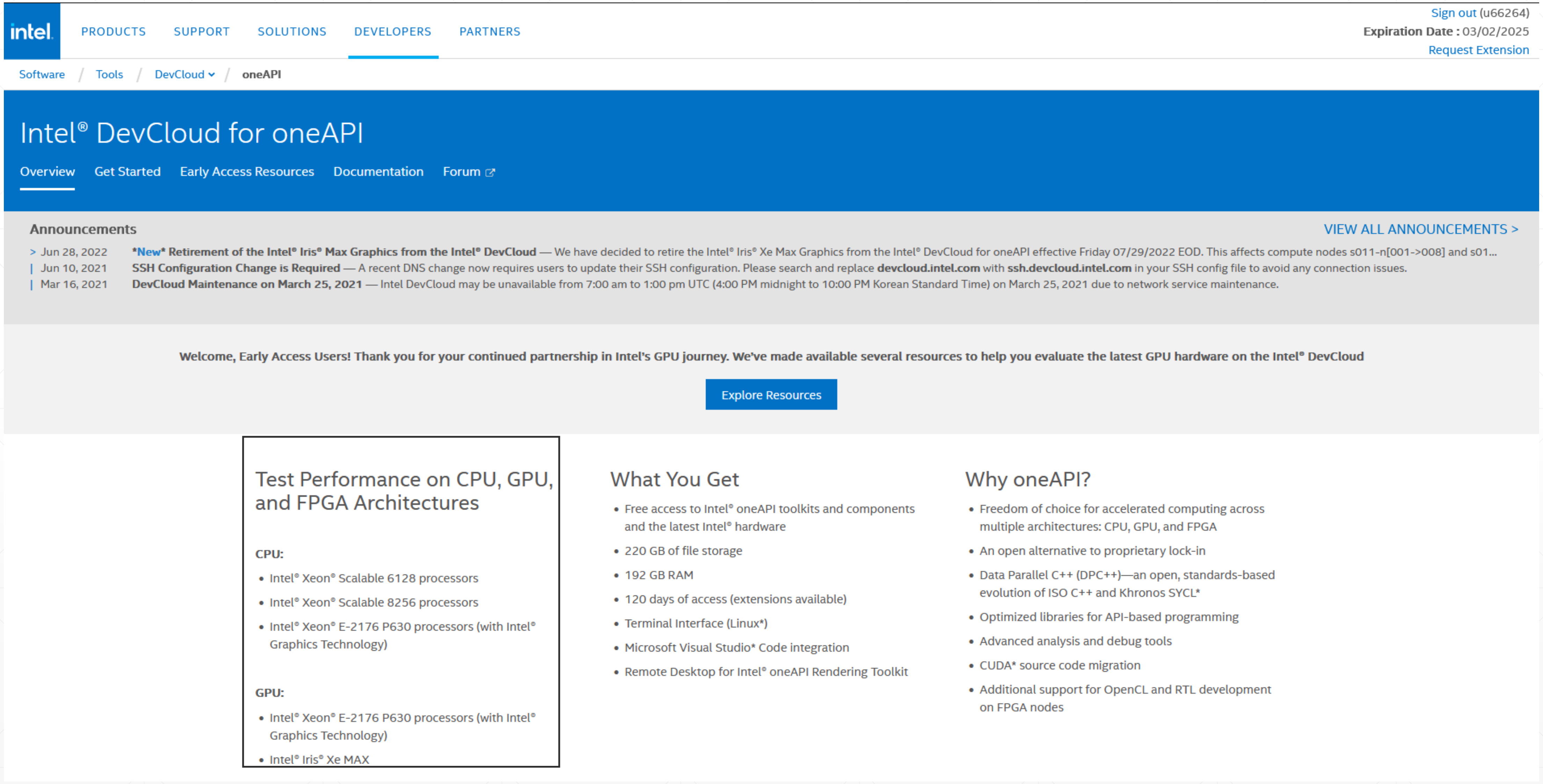
- Intel® Xeon® CPU Max Series: 1-node, 2x Intel® Xeon® CPU Max Series, HT On, Turbo On, Total Memory 128 GB HBM2e, BIOS EGSDCRB1.DWR.0085.D12.2207281916, ucode 0xac000040, SUSE Linux Enterprise Server 15 SP3, Kernel 5.3.18, oneAPI 2022.3.0, LAMMPS built with the Intel package
- Intel® Data Center GPU Max Series with DDR Host: 1-node, 2x Intel® Xeon® CPU Max Series, HT On, Turbo On, Total Memory 1024 GB DDR5-4800 + 128 GB HBM2e, Memory Mode: Flat, HBM2e not used, 6x Intel® Data Center GPU Max Series, BIOS EGSDCRB1.DWR.0085.D12.2207281916, ucode 0xac000040, Agama pvc-prq-54, SUSE Linux Enterprise Server 15 SP3, Kernel 5.3.18, oneAPI 2022.3.0, LAMMPS built with the Intel package
- Intel® Data Center GPU Max Series with HBM Host: 1-node, 2x Intel® Xeon® CPU Max Series, HT On, Turbo On, Total Memory 128 GB HBM2e, 6x Intel® Data Center GPU Max Series, BIOS EGSDCRB1.DWR.0085.D12.2207281916, ucode 0xac000040, Agama pvc-prq-54, SUSE Linux Enterprise Server 15 SP3, Kernel 5.3.18, oneAPI 2022.3.0, LAMMPS built with the Intel package.

See configuration disclosure for details. No product or component can be absolutely secure.

Performance varies by use, configuration, and other factors. Learn more at www.Intel.com/PerformanceIndex. Your costs and results may vary.



https://devcloud.intel.com/oneapi/



PRODUCTS SUPPORT SOLUTIONS DEVELOPERS PARTNERS

Sign out (u66264)
Expiration Date : 03/02/2025
Request Extension

Software / Tools / DevCloud / oneAPI

Intel® DevCloud for oneAPI

Overview Get Started Early Access Resources Documentation Forum

Announcements VIEW ALL ANNOUNCEMENTS >

> Jun 28, 2022 ***New* Retirement of the Intel® Iris® Max Graphics from the Intel® DevCloud** — We have decided to retire the Intel® Iris® Xe Max Graphics from the Intel® DevCloud for oneAPI effective Friday 07/29/2022 EOD. This affects compute nodes s011-n[001->008] and s01...
| Jun 10, 2021 **SSH Configuration Change is Required** — A recent DNS change now requires users to update their SSH configuration. Please search and replace **devcloud.intel.com** with **ssh.devcloud.intel.com** in your SSH config file to avoid any connection issues.
| Mar 16, 2021 **DevCloud Maintenance on March 25, 2021** — Intel DevCloud may be unavailable from 7:00 am to 1:00 pm UTC (4:00 PM midnight to 10:00 PM Korean Standard Time) on March 25, 2021 due to network service maintenance.

Welcome, Early Access Users! Thank you for your continued partnership in Intel's GPU journey. We've made available several resources to help you evaluate the latest GPU hardware on the Intel® DevCloud

Explore Resources

Test Performance on CPU, GPU, and FPGA Architectures

CPU:

- Intel® Xeon® Scalable 6128 processors
- Intel® Xeon® Scalable 8256 processors
- Intel® Xeon® E-2176 P630 processors (with Intel® Graphics Technology)

GPU:

- Intel® Xeon® E-2176 P630 processors (with Intel® Graphics Technology)
- Intel® Iris® Xe MAX

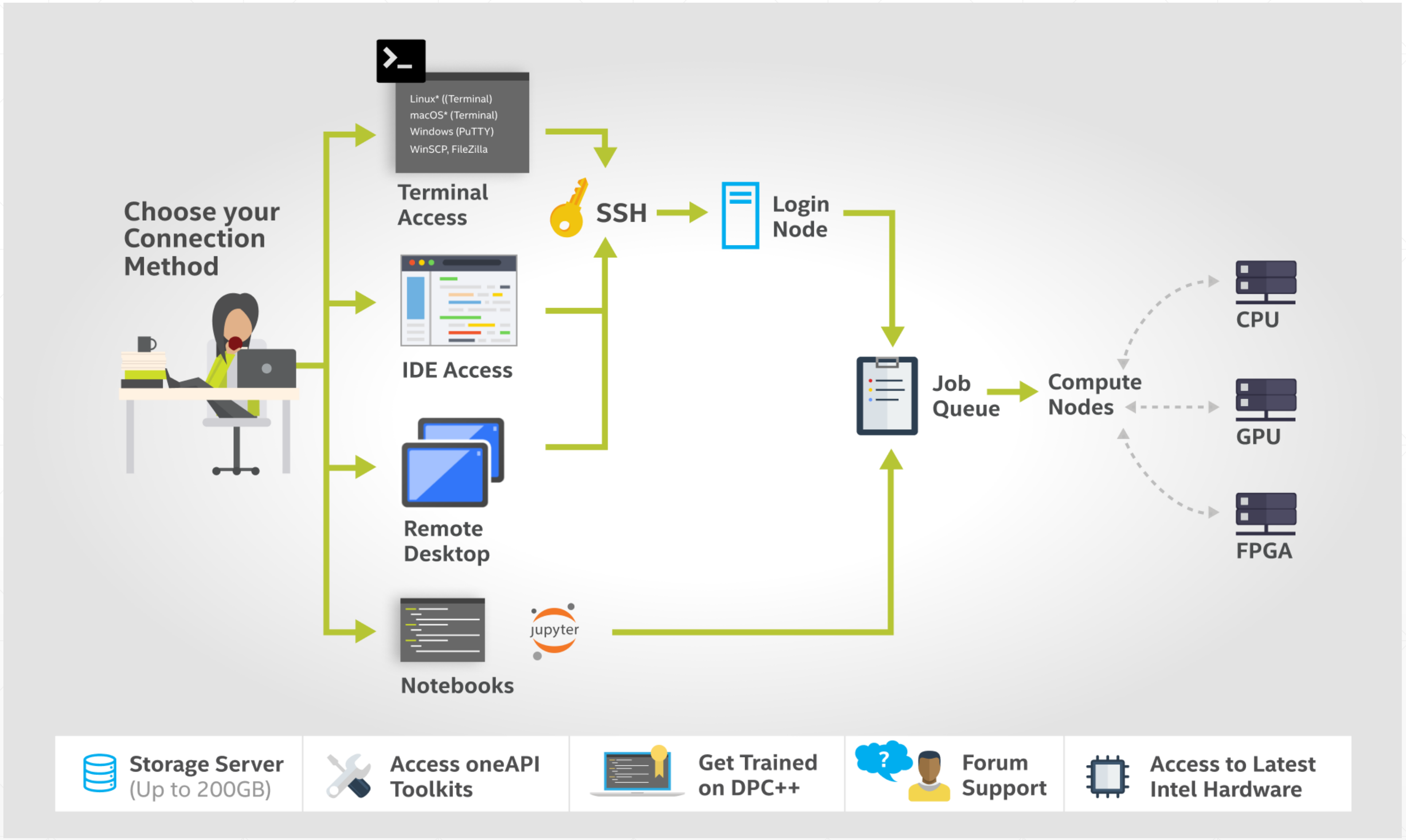
What You Get

- Free access to Intel® oneAPI toolkits and components and the latest Intel® hardware
- 220 GB of file storage
- 192 GB RAM
- 120 days of access (extensions available)
- Terminal Interface (Linux*)
- Microsoft Visual Studio* Code integration
- Remote Desktop for Intel® oneAPI Rendering Toolkit

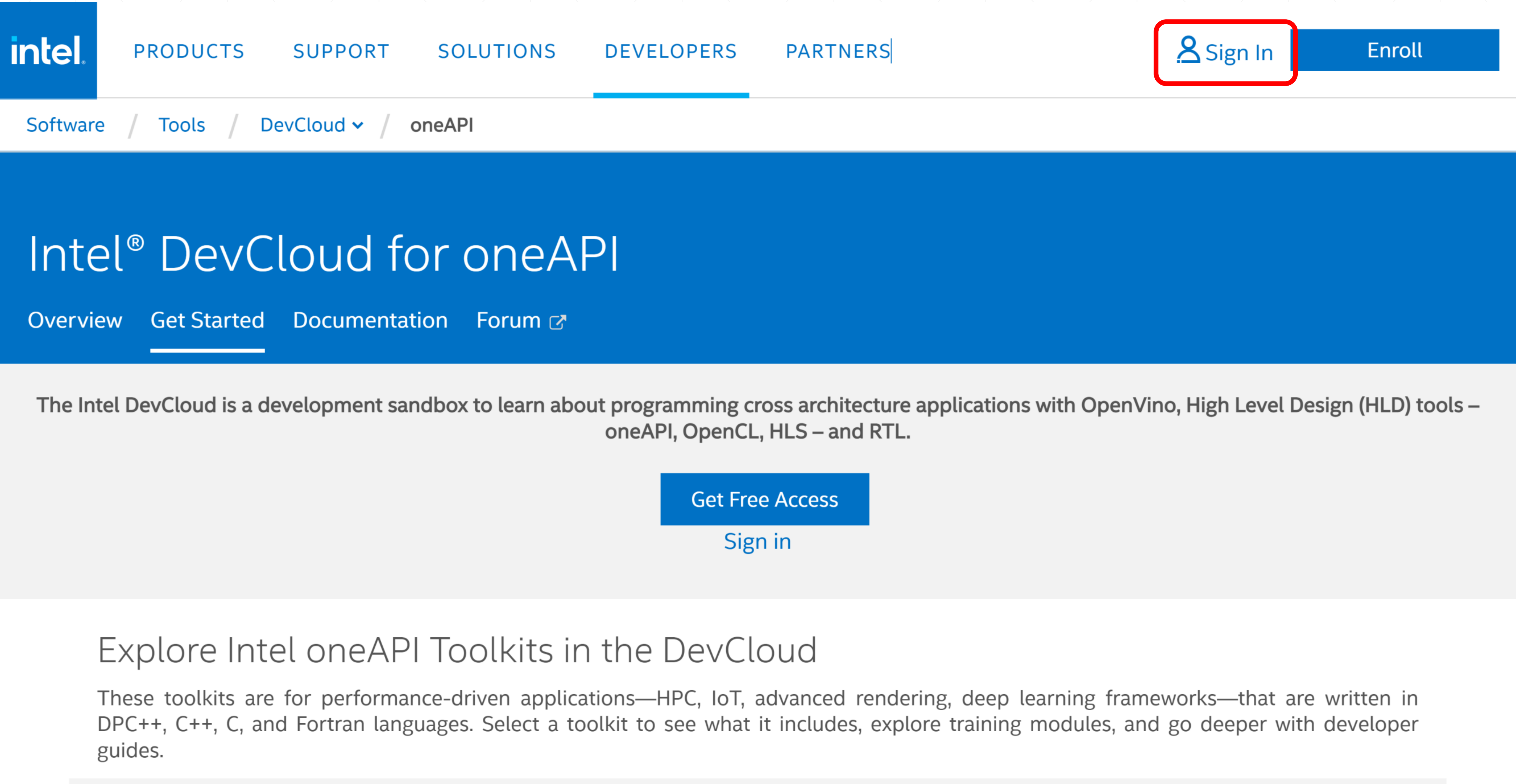
Why oneAPI?

- Freedom of choice for accelerated computing across multiple architectures: CPU, GPU, and FPGA
- An open alternative to proprietary lock-in
- Data Parallel C++ (DPC++)—an open, standards-based evolution of ISO C++ and Khronos SYCL*
- Optimized libraries for API-based programming
- Advanced analysis and debug tools
- CUDA* source code migration
- Additional support for OpenCL and RTL development on FPGA nodes

연결 방식


Choose your Connection Method
Linux* ((Terminal)
macOS* (Terminal)
Windows (PuTTY)
WinSCP, FileZilla
Terminal Access
IDE Access
Remote Desktop
Notebooks
jupyter
SSH
Login Node
Job Queue
Compute Nodes
CPU
GPU
FPGA
Storage Server (Up to 200GB)
Access oneAPI Toolkits
Get Trained on DPC++
Forum Support
Access to Latest Intel Hardware

Step 1) Visit https://devcloud.intel.com/oneapi/get_started/



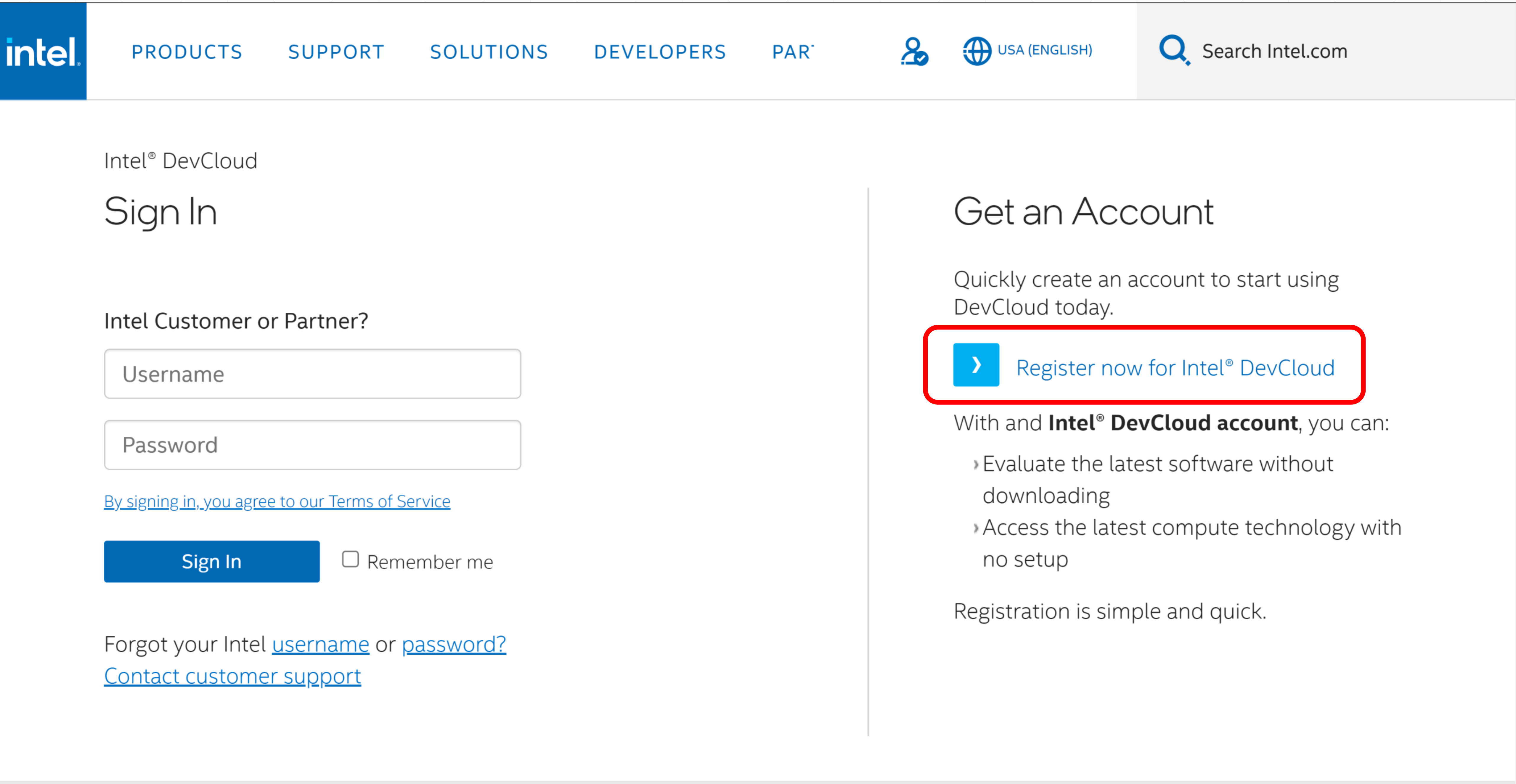
Intel® DevCloud for oneAPI

The Intel DevCloud is a development sandbox to learn about programming cross architecture applications with OpenVino, High Level Design (HLD) tools – oneAPI, OpenCL, HLS – and RTL.

Explore Intel oneAPI Toolkits in the DevCloud

These toolkits are for performance-driven applications—HPC, IoT, advanced rendering, deep learning frameworks—that are written in DPC++, C++, C, and Fortran languages. Select a toolkit to see what it includes, explore training modules, and go deeper with developer guides.

Step 2) Click the "Register now for Intel® DevCloud" link



PRODUCTS SUPPORT SOLUTIONS DEVELOPERS PAR

USA (ENGLISH)

Search Intel.com

Intel® DevCloud

Sign In

Intel Customer or Partner?

Username

Password

By signing in, you agree to our Terms of Service

Sign In

☐ Remember me

Forgot your Intel username or password?

Contact customer support

Get an Account

Quickly create an account to start using DevCloud today.

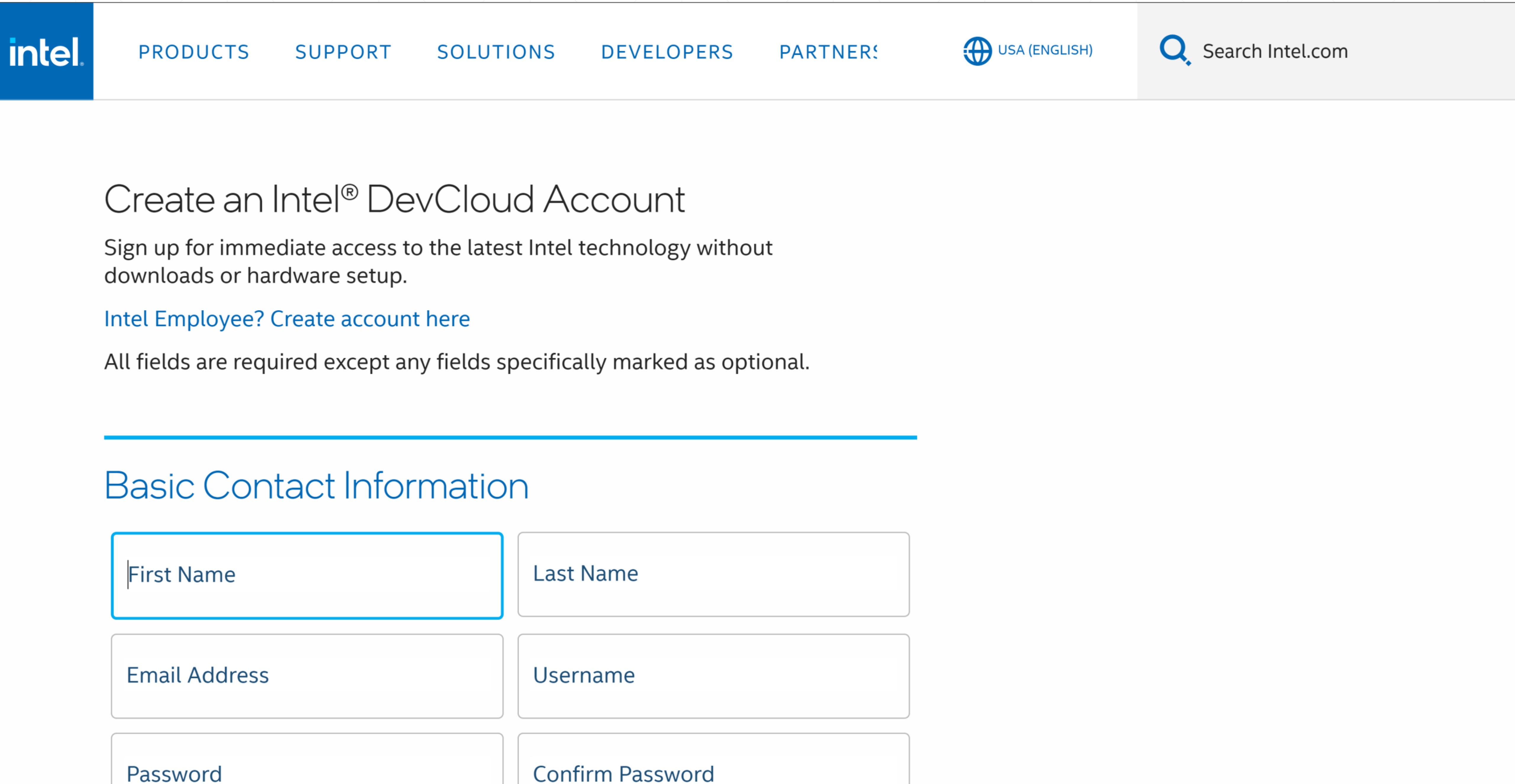
› Register now for Intel® DevCloud

With and **Intel® DevCloud account**, you can:

- › Evaluate the latest software without downloading
- › Access the latest compute technology with no setup

Registration is simple and quick.

Step 3) Fill out the “Basic Contact Information” section



Create an Intel® DevCloud Account

Sign up for immediate access to the latest Intel technology without downloads or hardware setup.

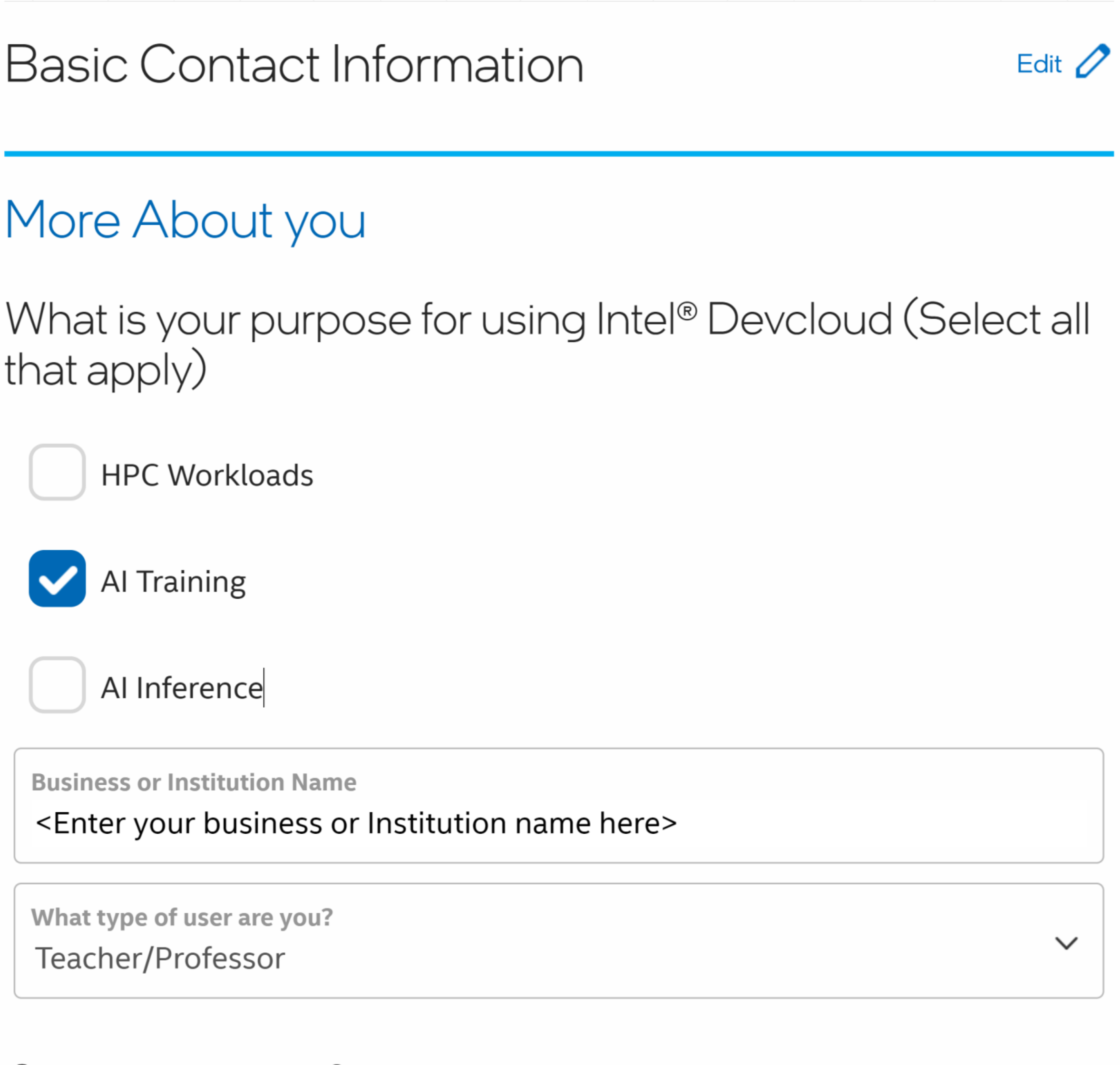
Intel Employee? Create account here

All fields are required except any fields specifically marked as optional.

Basic Contact Information

First Name	Last Name
Email Address	Username
Password	Confirm Password

Step 4) Fill out the “More About you” section



Basic Contact Information

Edit

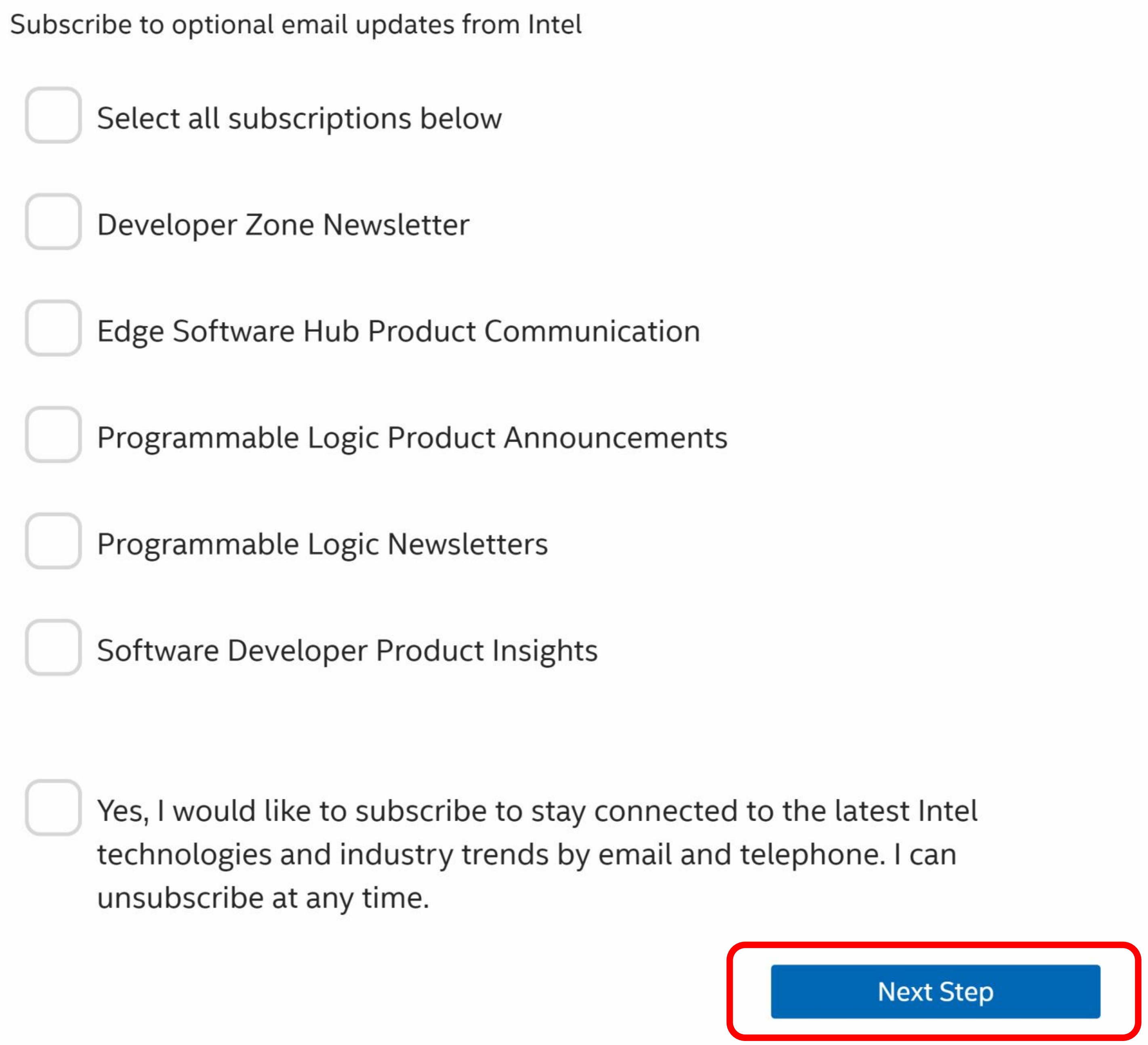
More About you

What is your purpose for using Intel® Devcloud (Select all that apply)

- [ ] HPC Workloads
- [x] AI Training
- [ ] AI Inference

Business or Institution Name	<Enter your business or Institution name here>
What type of user are you?	Teacher/Professor

Step 5) Select what applies and click the "Next Step" button

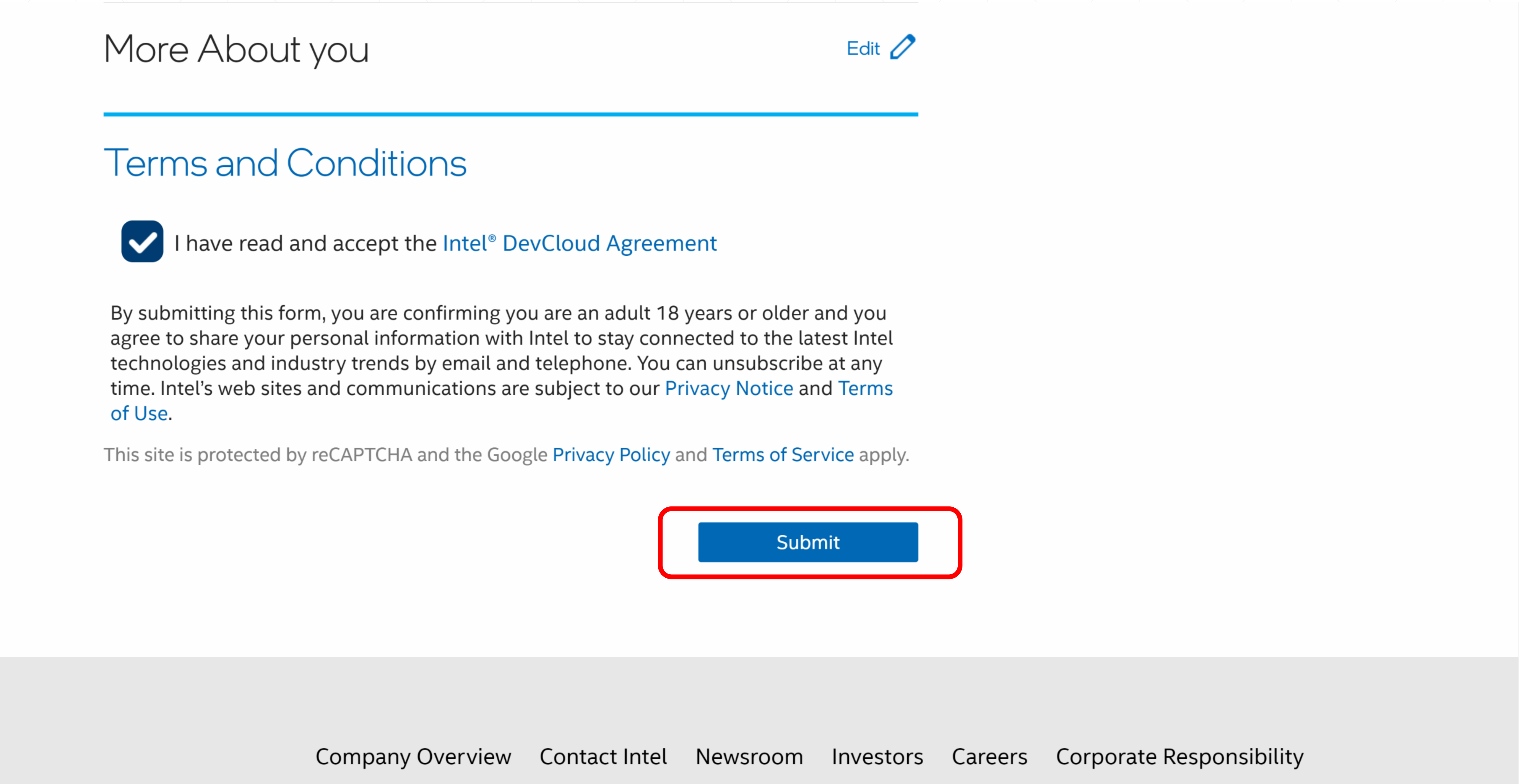


Subscribe to optional email updates from Intel

- ☐ Select all subscriptions below
- ☐ Developer Zone Newsletter
- ☐ Edge Software Hub Product Communication
- ☐ Programmable Logic Product Announcements
- ☐ Programmable Logic Newsletters
- ☐ Software Developer Product Insights

☐ Yes, I would like to subscribe to stay connected to the latest Intel technologies and industry trends by email and telephone. I can unsubscribe at any time.

Step 6) Accept the Terms and Conditions and click the "Submit" button



More About you

Edit

Terms and Conditions

✓ I have read and accept the Intel® DevCloud Agreement

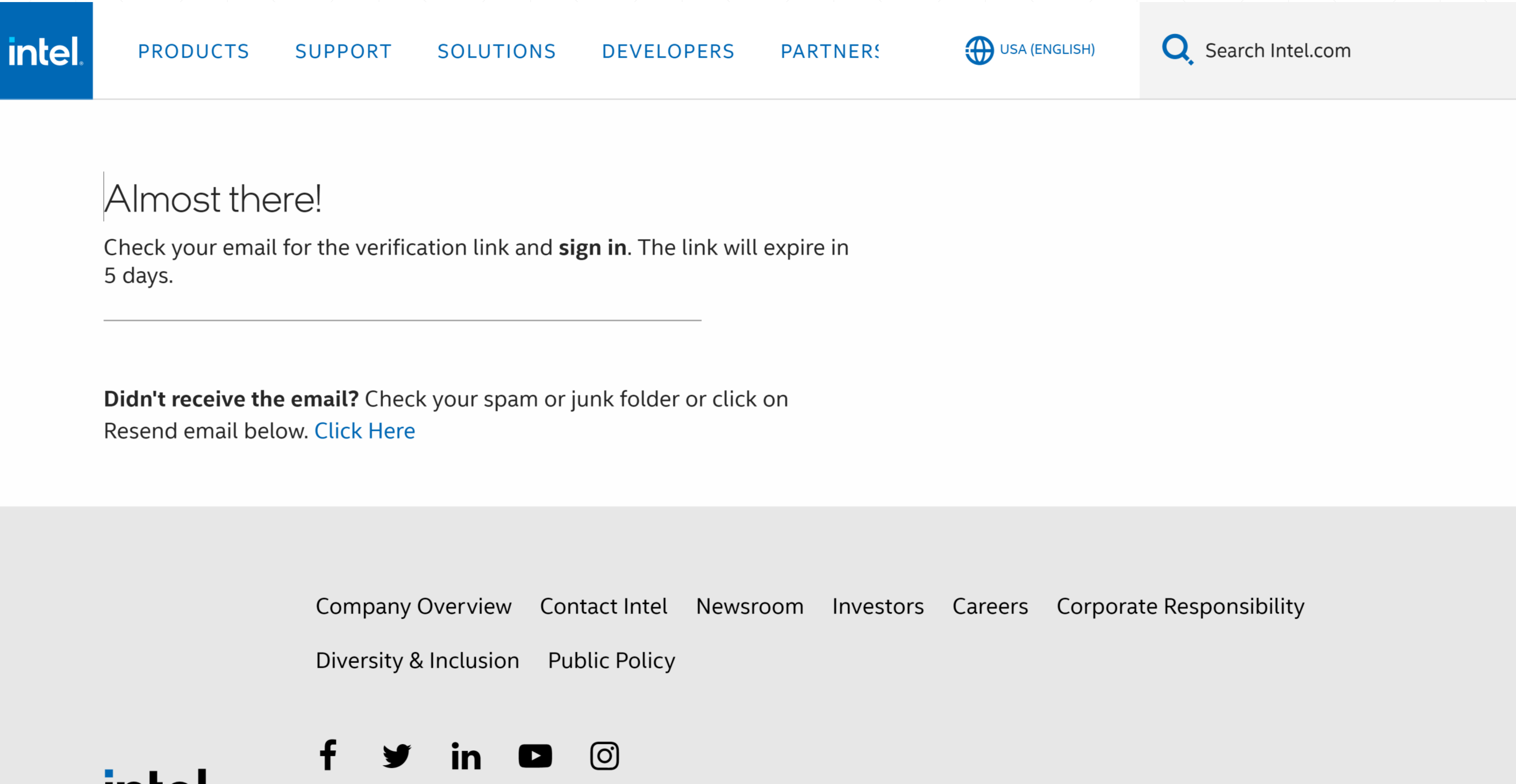
By submitting this form, you are confirming you are an adult 18 years or older and you agree to share your personal information with Intel to stay connected to the latest Intel technologies and industry trends by email and telephone. You can unsubscribe at any time. Intel's web sites and communications are subject to our Privacy Notice and Terms of Use.

This site is protected by reCAPTCHA and the Google Privacy Policy and Terms of Service apply.

Submit

Company Overview Contact Intel Newsroom Investors Careers Corporate Responsibility

Step 7) Check your email to verify your email address

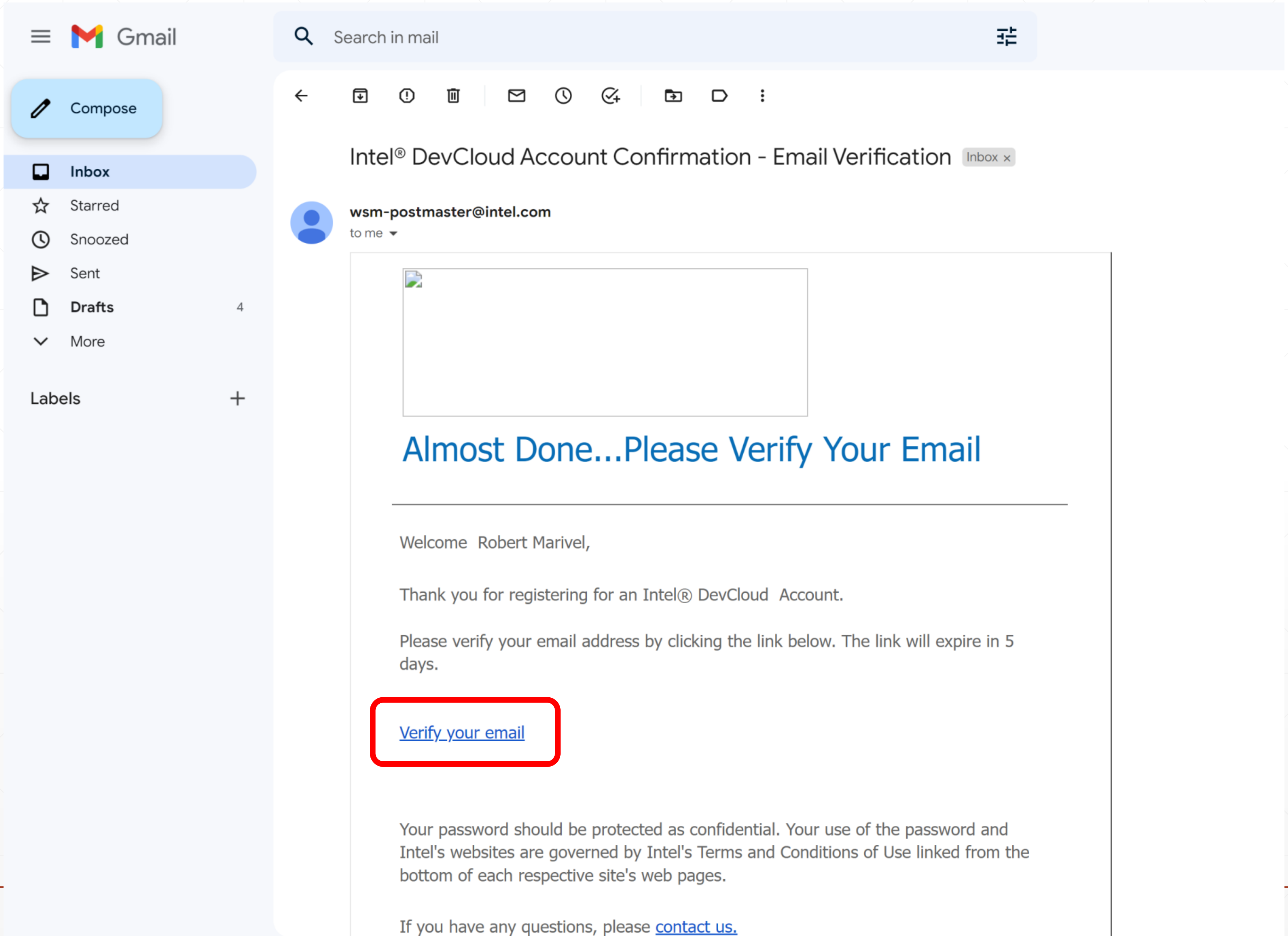


Almost there!

Check your email for the verification link and **sign in**. The link will expire in 5 days.

Didn't receive the email? Check your spam or junk folder or click on Resend email below. Click Here

Step 8) Click the "Verify your email" link in the email sent by wsm-postmaster@intel.com



Intel® DevCloud Account Confirmation - Email Verification

wsm-postmaster@intel.com
to me

Almost Done...Please Verify Your Email

Welcome Robert Marivel,

Thank you for registering for an Intel® DevCloud Account.

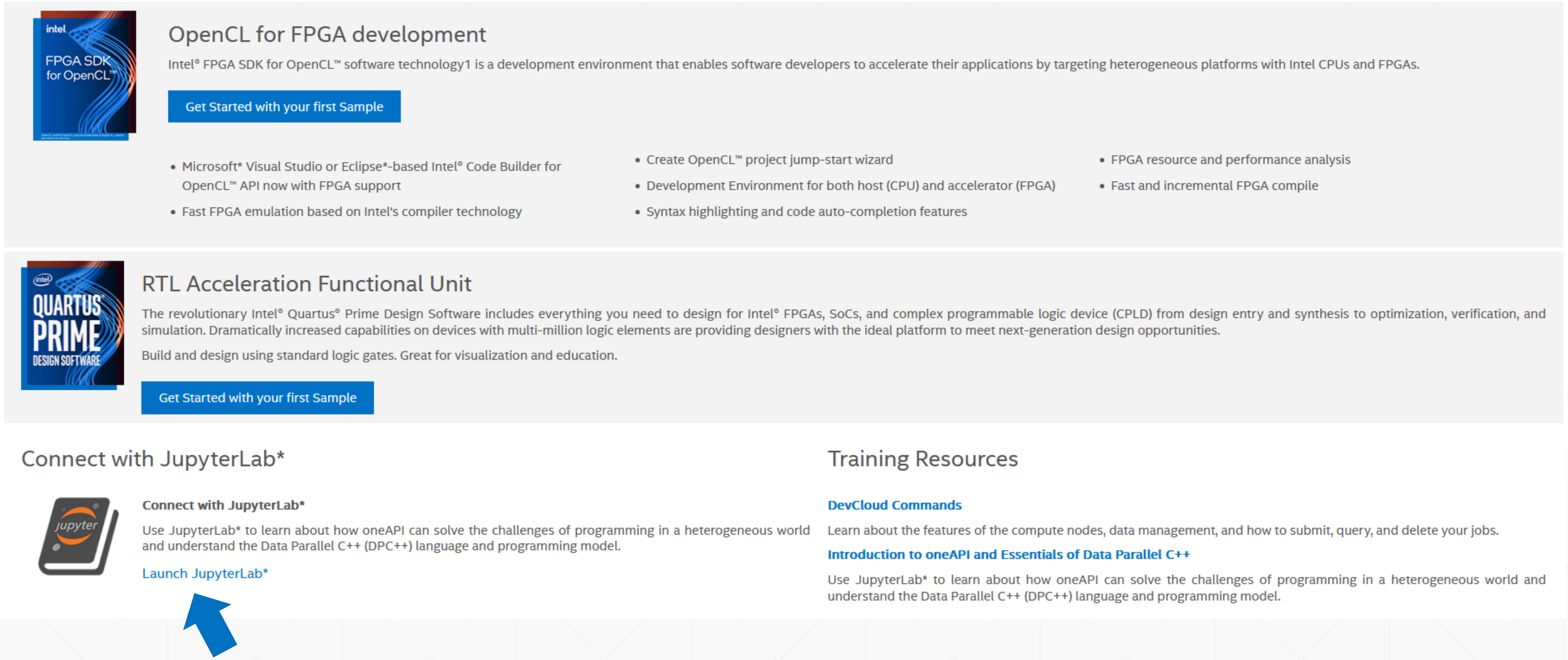
Please verify your email address by clicking the link below. The link will expire in 5 days.

Verify your email

Your password should be protected as confidential. Your use of the password and Intel's websites are governed by Intel's Terms and Conditions of Use linked from the bottom of each respective site's web pages.

If you have any questions, please contact us.

https://devcloud.intel.com/oneapi/get_started/



OpenCL for FPGA development

Intel® FPGA SDK for OpenCL™ software technology1 is a development environment that enables software developers to accelerate their applications by targeting heterogeneous platforms with Intel CPUs and FPGAs.

Get Started with your first Sample

- Microsoft* Visual Studio or Eclipse*-based Intel® Code Builder for OpenCL™ API now with FPGA support
- Fast FPGA emulation based on Intel's compiler technology
- Create OpenCL™ project jump-start wizard
- Development Environment for both host (CPU) and accelerator (FPGA)
- Syntax highlighting and code auto-completion features
- FPGA resource and performance analysis
- Fast and incremental FPGA compile

RTL Acceleration Functional Unit

The revolutionary Intel® Quartus® Prime Design Software includes everything you need to design for Intel® FPGAs, SoCs, and complex programmable logic device (CPLD) from design entry and synthesis to optimization, verification, and simulation. Dramatically increased capabilities on devices with multi-million logic elements are providing designers with the ideal platform to meet next-generation design opportunities.

Build and design using standard logic gates. Great for visualization and education.

Get Started with your first Sample

Connect with JupyterLab*

Connect with JupyterLab*

Use JupyterLab* to learn about how oneAPI can solve the challenges of programming in a heterogeneous world and understand the Data Parallel C++ (DPC++) language and programming model.

Launch JupyterLab*

Training Resources

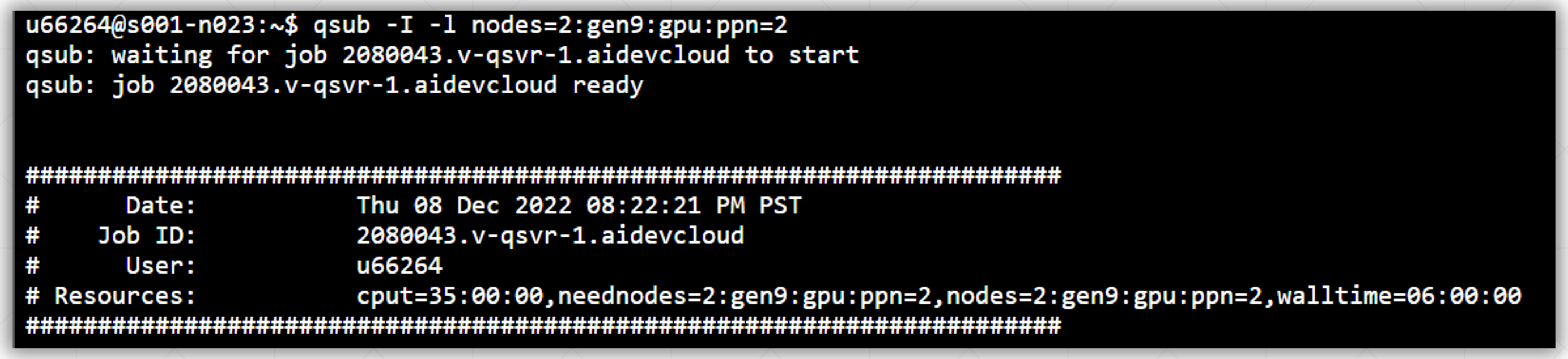
DevCloud Commands

Learn about the features of the compute nodes, data management, and how to submit, query, and delete your jobs.

Introduction to oneAPI and Essentials of Data Parallel C++

Use JupyterLab* to learn about how oneAPI can solve the challenges of programming in a heterogeneous world and understand the Data Parallel C++ (DPC++) language and programming model.

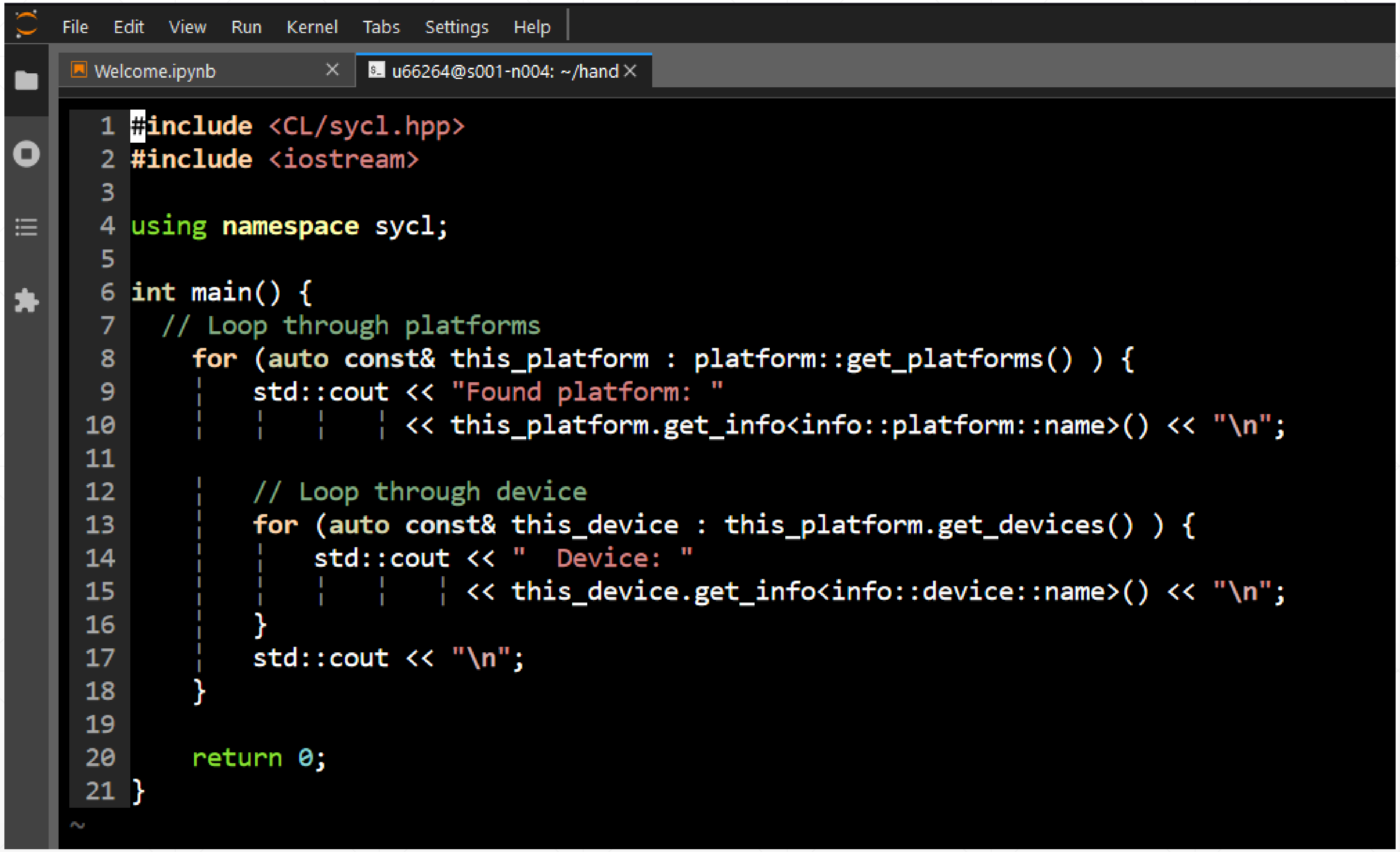
인터랙티브 작업 재출



```
u66264@s001-n023:~$ qsub -I -l nodes=2:gen9:gpu:ppn=2
qsub: waiting for job 2080043.v-qsvr-1.aidevcloud to start
qsub: job 2080043.v-qsvr-1.aidevcloud ready


########################################################################
#      Date:            Thu 08 Dec 2022 08:22:21 PM PST
#    Job ID:            2080043.v-qsvr-1.aidevcloud
#      User:            u66264
# Resources:            cput=35:00:00,neednodes=2:gen9:gpu:ppn=2,nodes=2:gen9:gpu:ppn=2,walltime=06:00:00
########################################################################
```

- Request node based on device properties
  - qsub -I -l nodes=[nnodes]:[props]:ppn=[process_per_node]
- Properties describing device class:
  - core / xeon / gpu / fpga
- Properties describing device name:
  - gen9 / gen 11 / aria10 / stratix10 / gold6128 / i9-10920x
- Properties describing purpose:
  - fpga_compile / fpga_runtime / renderkit

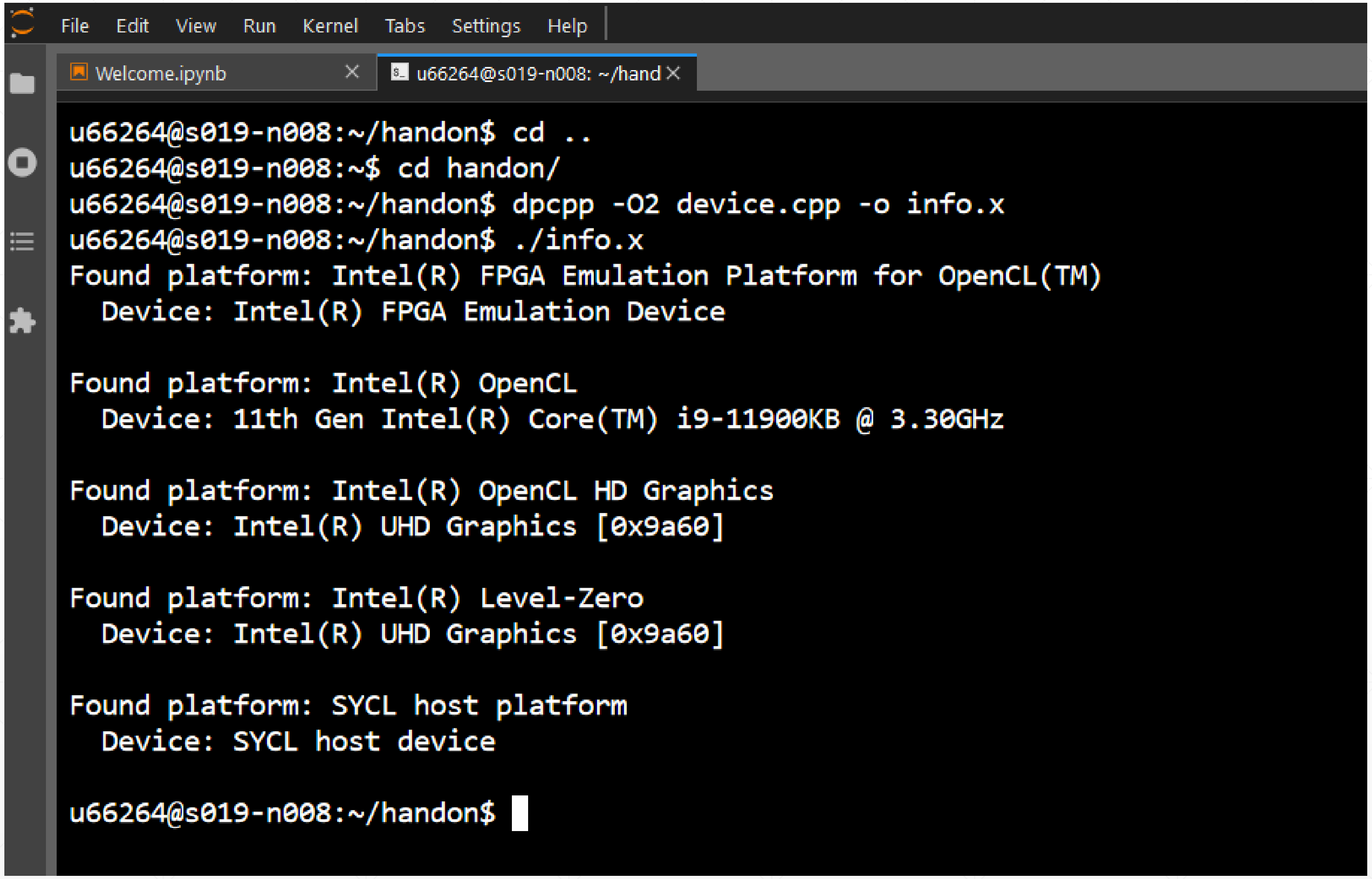
디바이스 및 플랫폼 발견



```cpp
#include <CL/sycl.hpp>
#include <iostream>

using namespace sycl;

int main() {
  // Loop through platforms
    for (auto const& this_platform : platform::get_platforms() ) {
        std::cout << "Found platform: "
                  << this_platform.get_info<info::platform::name>() << "\n";

        // Loop through device
        for (auto const& this_device : this_platform.get_devices() ) {
            std::cout << "  Device: "
                      << this_device.get_info<info::device::name>() << "\n";
        }
        std::cout << "\n";
    }

    return 0;
}
```

https://github.com/Apress/data-parallel-CPP/tree/main/samples/Ch12_device_information

Intel® Gen 9 그래픽들



```
u66264@s019-n008:~/handon$ cd ..
u66264@s019-n008:~$ cd handon/
u66264@s019-n008:~/handon$ dpcpp -O2 device.cpp -o info.x
u66264@s019-n008:~/handon$ ./info.x
Found platform: Intel(R) FPGA Emulation Platform for OpenCL(TM)
  Device: Intel(R) FPGA Emulation Device

Found platform: Intel(R) OpenCL
  Device: 11th Gen Intel(R) Core(TM) i9-11900KB @ 3.30GHz

Found platform: Intel(R) OpenCL HD Graphics
  Device: Intel(R) UHD Graphics [0x9a60]

Found platform: Intel(R) Level-Zero
  Device: Intel(R) UHD Graphics [0x9a60]

Found platform: SYCL host platform
  Device: SYCL host device

u66264@s019-n008:~/handon$
```

SYCL Hands-on: Vector Add

```
#include <vector>
#include <iostream>
#include <string>

#include <sycl/sycl.hpp>
#if FPGA_EMULATOR
#include <sycl/ext/intel/fpga_extensions.hpp>
#endif

int main(int argc, char* argv[]) {
  // vector size
  auto size = 10000;
  std::vector<int> a(size), b(size), sum_host(size), sum_device(size)

  // vector initialization
  for (auto i = 0; i < a.size(); i++) a.at(i) = i;
  for (auto i = 0; i < b.size(); i++) b.at(i) = 2*i;

  // Compute the sum of two vectors in sequential for validation.
  for (auto i = 0; i < sum_host.size(); i++)
    sum_host.at(i) = a.at(i) + b.at(i);
```

SYCL Hands-on: 디바이스 선택 및 정보

```
#if FPGA_EMULATOR
  // FPGA emulator selector on systems without FPGA card.
  auto selector = sycl::ext::intel::fpga_emulator_selector_v;
#else
  // The default  is the most performant device.
  auto selector = default_selector_v;
#endif

// Create SYCL queue tied to device
queue q(selector);

// Print out the device information used for the kernel code.
std::cout << "Running on device: "
          << q.get_device().get_info<info::device::name>() << "₩n";

std::cout << "Vector size: " << a.size() << "₩n";
```

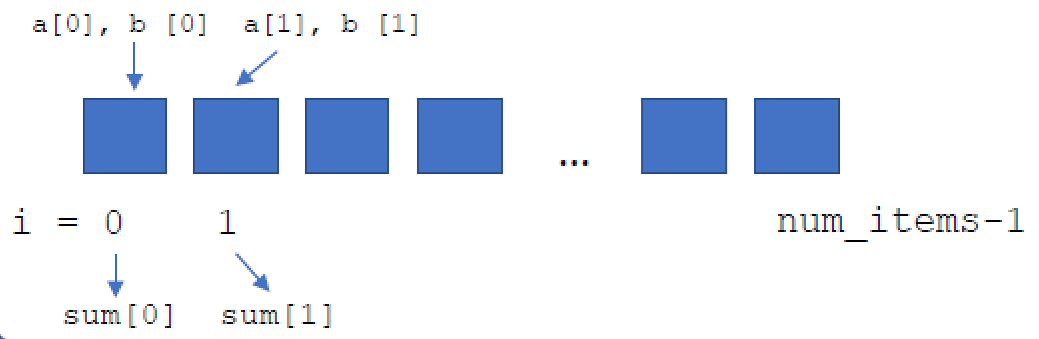
SYCL Hands-on: 호스트 vs. 디바이스 코드

`for_loop`: Iterate over data using the same operation:

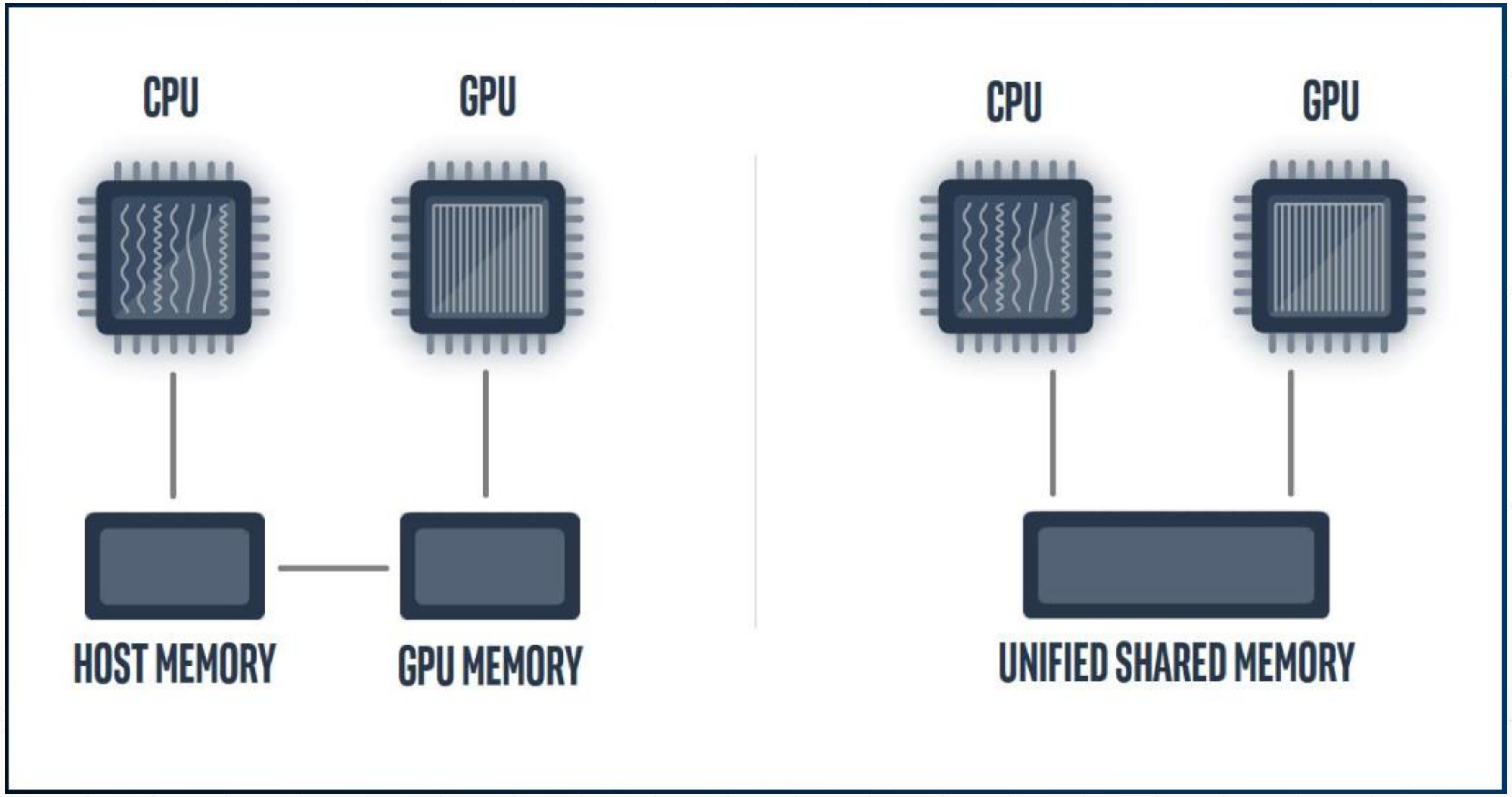
```
For (int i = 0; < num_times;
 ++i) {sum[i] = a[i] + b[i];
};
```

`kernel`: Launch `num_items` kernel instances (work items) to be executed in parallel. Each instance uses separate pieces of data. `i` is a unique identifier in the execution range 0 … `num_items-1`:

```
h.parallel_for(num_items, [=](auto i) { sum[i] = a[i] +
b[i]; });
```



SYCL Hands-on: Buffer vs. USM



SYCL Hands-on: Buffer 이용

```
// Create buffers sharing data between the host and the devices.
buffer a_buf(a_vector);
buffer b_buf(b_vector);
buffer sum_buf(sum_device.data(), num_items);

 // Submit kernel code to device
 q.submit([&](handler &h) {
    // Create an accessor for input buffer with read permissions
    accessor a(a_buf, h, read_only);
    accessor b(b_buf, h, read_only);

    // Create an accessor for output buffer with write permissions
    accessor sum(sum_buf, h, write_only);

    h.parallel_for(size, [=](auto i) { sum[i] = a[i] + b[i]; });
 });
 // data is copied back when buffer goes out of scope
```

SYCL Hands-on: 결과 확인

```
  // Verify that the two vectors are equal.
  for (auto i = 0; i < sum_host.size(); i++) {
    if (sum_host.at(i) != sum_device.at(i)) {
      std::cout << "Vector add failed on device.₩n";
      return -1;
    }
  }

  // Free memories
  a.clear();
  b.clear();
  sum_host.clear();
  sum_device.clear();

  std::cout << "Vector add successfully completed on device.₩n";

  return 0;
}
```

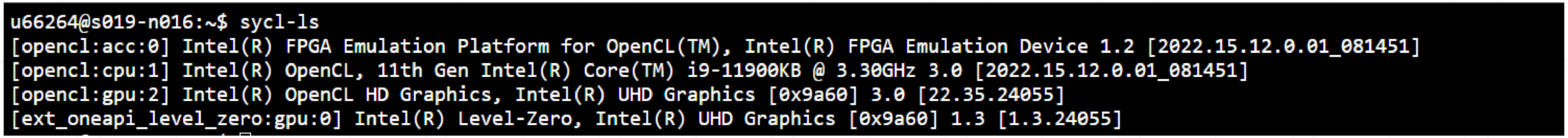
SYCL Hands-on: USM 이용

```
// unified shared memory so that both CPU and device can access them.
int *a = malloc_shared<int>(size, q);
int *b = malloc_shared<int>(size, q);
int *sum_host = malloc_shared<int>(array_size, q);
int *sum_host = malloc_shared<int>(array_size, q);

// vector initialization
for (auto i = 0; i < size; i++) a[i] = i;
for (auto i = 0; i < size; i++) b[i] = 2*i;

// Compute the sum of two vectors in sequential for validation.
for (auto i = 0; i < size; i++)
  sum_host[i] = a[i] + b[i];

// Implicit submit for simple kernel
auto e = q.parallel_for(size, [=](auto i) {
  sum[i] = a[i] + b[i];
});
```

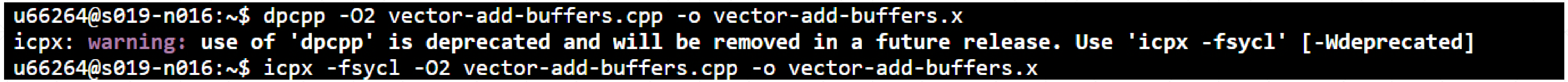
SYCL Hands-on: 결과 확인

```
  // Verify that the two vectors are equal.
  for (auto i = 0; i < size; i++) {
    if (sum_host[i] != sum_device[i]) {
      std::cout << "Vector add failed on device.₩n";
      return -1;
    }
  }

  // Free memories
  free(a, q);
  free(b, q);
  free(sum_sequential, q);
  free(sum_parallel, q);

  std::cout << "Vector add successfully completed on device.₩n";

  return 0;
}
```

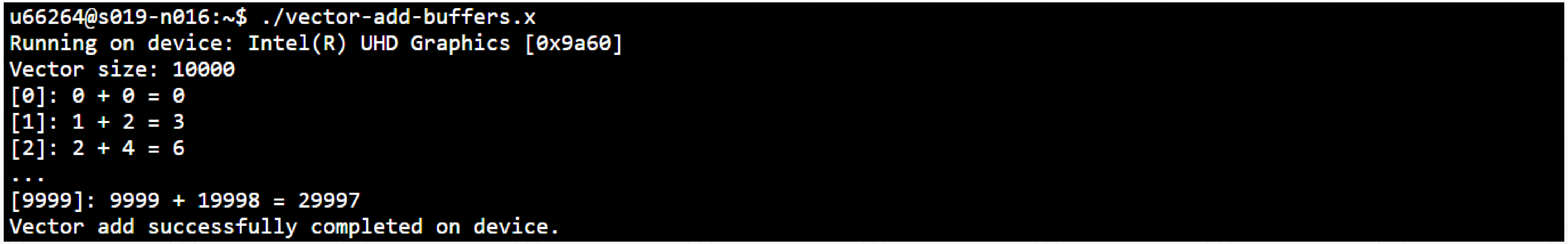
SYCL Hands-on: 컴파일 및 실행 테스트



```
u66264@s019-n016:~$ sycl-ls
[opencl:acc:0] Intel(R) FPGA Emulation Platform for OpenCL(TM), Intel(R) FPGA Emulation Device 1.2 [2022.15.12.0.01_081451]
[opencl:cpu:1] Intel(R) OpenCL, 11th Gen Intel(R) Core(TM) i9-11900KB @ 3.30GHz 3.0 [2022.15.12.0.01_081451]
[opencl:gpu:2] Intel(R) OpenCL HD Graphics, Intel(R) UHD Graphics [0x9a60] 3.0 [22.35.24055]
[ext_oneapi_level_zero:gpu:0] Intel(R) Level-Zero, Intel(R) UHD Graphics [0x9a60] 1.3 [1.3.24055]
```

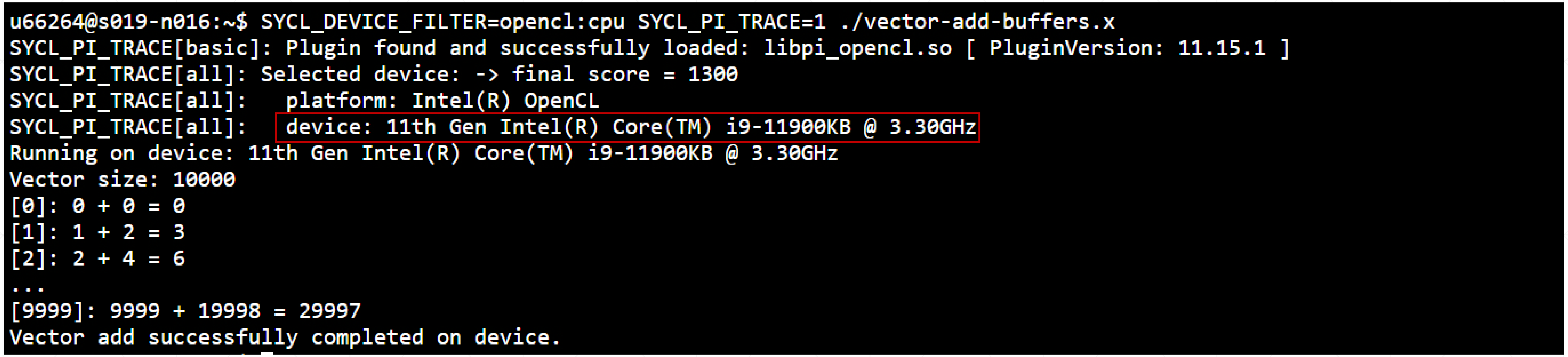


```
u66264@s019-n016:~$ dpcpp -O2 vector-add-buffers.cpp -o vector-add-buffers.x
icpx: warning: use of 'dpcpp' is deprecated and will be removed in a future release. Use 'icpx -fsycl' [-Wdeprecated]
u66264@s019-n016:~$ icpx -fsycl -O2 vector-add-buffers.cpp -o vector-add-buffers.x
```

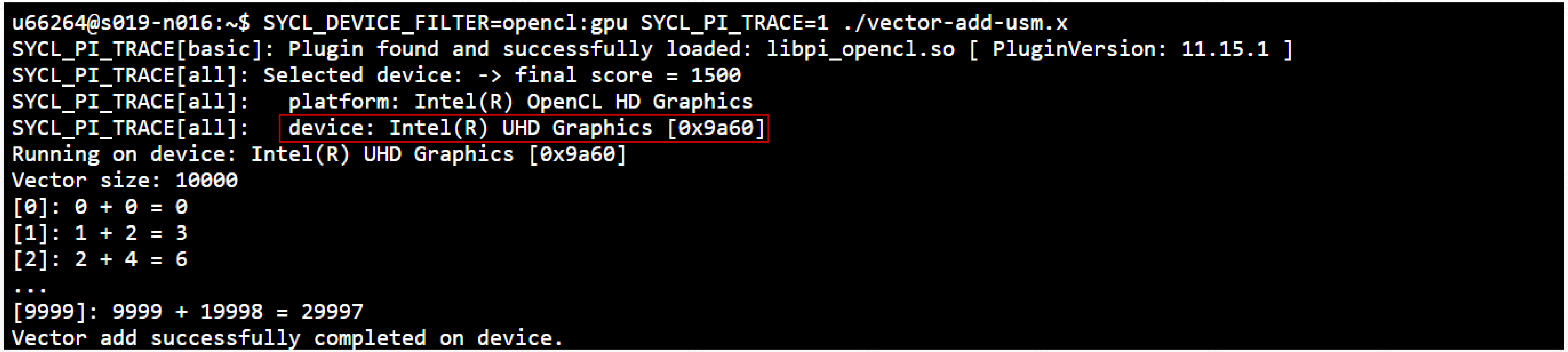


```
u66264@s019-n016:~$ ./vector-add-buffers.x
Running on device: Intel(R) UHD Graphics [0x9a60]
Vector size: 10000
[0]: 0 + 0 = 0
[1]: 1 + 2 = 3
[2]: 2 + 4 = 6
...
[9999]: 9999 + 19998 = 29997
Vector add successfully completed on device.
```

SYCL Hands-on: 실행 시 디바이스 선택



```
u66264@s019-n016:~$ SYCL_DEVICE_FILTER=opencl:cpu SYCL_PI_TRACE=1 ./vector-add-buffers.x
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_opencl.so [ PluginVersion: 11.15.1 ]
SYCL_PI_TRACE[all]: Selected device: -> final score = 1300
SYCL_PI_TRACE[all]:   platform: Intel(R) OpenCL
SYCL_PI_TRACE[all]:   device: 11th Gen Intel(R) Core(TM) i9-11900KB @ 3.30GHz
Running on device: 11th Gen Intel(R) Core(TM) i9-11900KB @ 3.30GHz
Vector size: 10000
[0]: 0 + 0 = 0
[1]: 1 + 2 = 3
[2]: 2 + 4 = 6
...
[9999]: 9999 + 19998 = 29997
Vector add successfully completed on device.
```



```
u66264@s019-n016:~$ SYCL_DEVICE_FILTER=opencl:gpu SYCL_PI_TRACE=1 ./vector-add-usm.x
SYCL_PI_TRACE[basic]: Plugin found and successfully loaded: libpi_opencl.so [ PluginVersion: 11.15.1 ]
SYCL_PI_TRACE[all]: Selected device: -> final score = 1500
SYCL_PI_TRACE[all]:   platform: Intel(R) OpenCL HD Graphics
SYCL_PI_TRACE[all]:   device: Intel(R) UHD Graphics [0x9a60]
Running on device: Intel(R) UHD Graphics [0x9a60]
Vector size: 10000
[0]: 0 + 0 = 0
[1]: 1 + 2 = 3
[2]: 2 + 4 = 6
...
[9999]: 9999 + 19998 = 29997
Vector add successfully completed on device.
```

SYCL Hands-on: CUDA

```
__global__ void VectorAddKernel(float* A, float* B, float* C)
{
    A[threadIdx.x] = threadIdx.x + 1.0f;
    B[threadIdx.x] = threadIdx.x + 1.0f;
    C[threadIdx.x] = A[threadIdx.x] + B[threadIdx.x];
}

int main()
{
    float *d_A, *d_B, *d_C;
    cudaError_t status;

    cudaMalloc(&d_A, VECTOR_SIZE*sizeof(float));
    cudaMalloc(&d_B, VECTOR_SIZE*sizeof(float));
    cudaMalloc(&d_C, VECTOR_SIZE*sizeof(float));

    VectorAddKernel<<<1, VECTOR_SIZE>>>(d_A, d_B, d_C);
```

SYCL Hands-on: CUDA

```
  status = cudaMemcpy(Result, d_C, SIZE*sizeof(float), cudaMemcpyDeviceToHost);
  if (status != cudaSuccess) {
    printf("Could not copy result to host₩n");
    exit(EXIT_FAILURE);
  }

  cudaFree(d_A);
  cudaFree(d_B);
  cudaFree(d_C);

  for (int i = 0; i < VECTOR_SIZE; i++) {
    if (i % 16 == 0) {
      printf("₩n");
    }
    printf("%3.0f ", Result[i]);
  }
  printf("₩n");

  return 0;
}
```

SYCL Hands-on: 호환성 도구 옵션

--help	A list of dpct specific options.
--in-root=<dir>	Directory path for the root of the source tree to be migrated
--out-root=<dir>	Directory path for root of generated files.

--assume-nd-range-dim=<value>
  =1
  =3
	Dimensionality of *nd_range* to use in generated code
	  1D nd_range where possible, and 3D in other cases
	  3d nd_range (default)

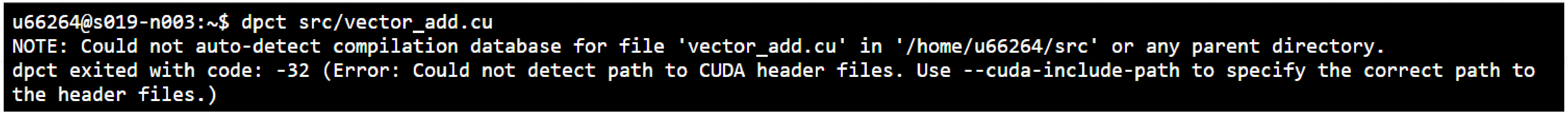
--comments	Comments explaining the generated code. (Default: off)

--output-verbosity=<value>
  =silent
  =normal
  =detailed
  =diagnostics
	Sets the output verbosity level:
	  Only messages from clang.
	  'silent' and warnings, errors, and notes from dpct.
	  'normal' and messages about which file is being processed.
	  'detailed' and information about the conflicts and crashes

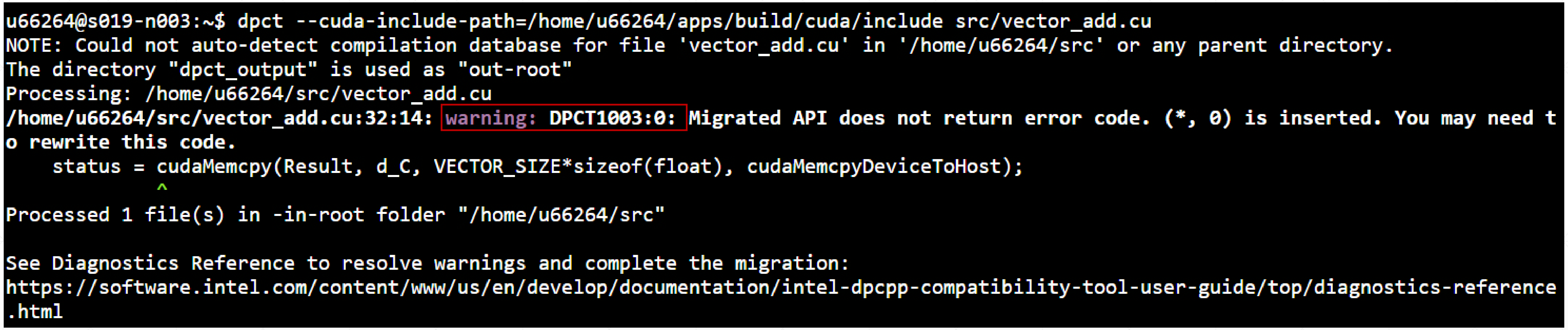
--suppress-warnings=<value>	Comma separated list of migration warnings to suppress
--cuda-include-path=<dir>	Directory path of the CUDA header files
--enable-ctad	C++17 class template argument deduction (CTAD)

--usm-level=<value>
  =restricted
  =none
	Unified Shared Memory (USM) level in source code generation.
	  Uses USM API for memory management migration. (default)
	  Uses helper functions from DPCT header files

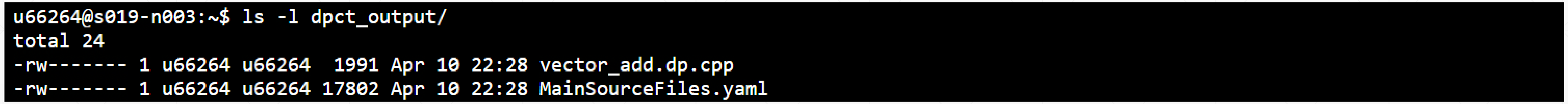
SYCL Hands-on: CUDA 헤더파일 관련 오류



```
u66264@s019-n003:~$ dpct src/vector_add.cu
NOTE: Could not auto-detect compilation database for file 'vector_add.cu' in '/home/u66264/src' or any parent directory.
dpct exited with code: -32 (Error: Could not detect path to CUDA header files. Use --cuda-include-path to specify the correct path to
the header files.)
```



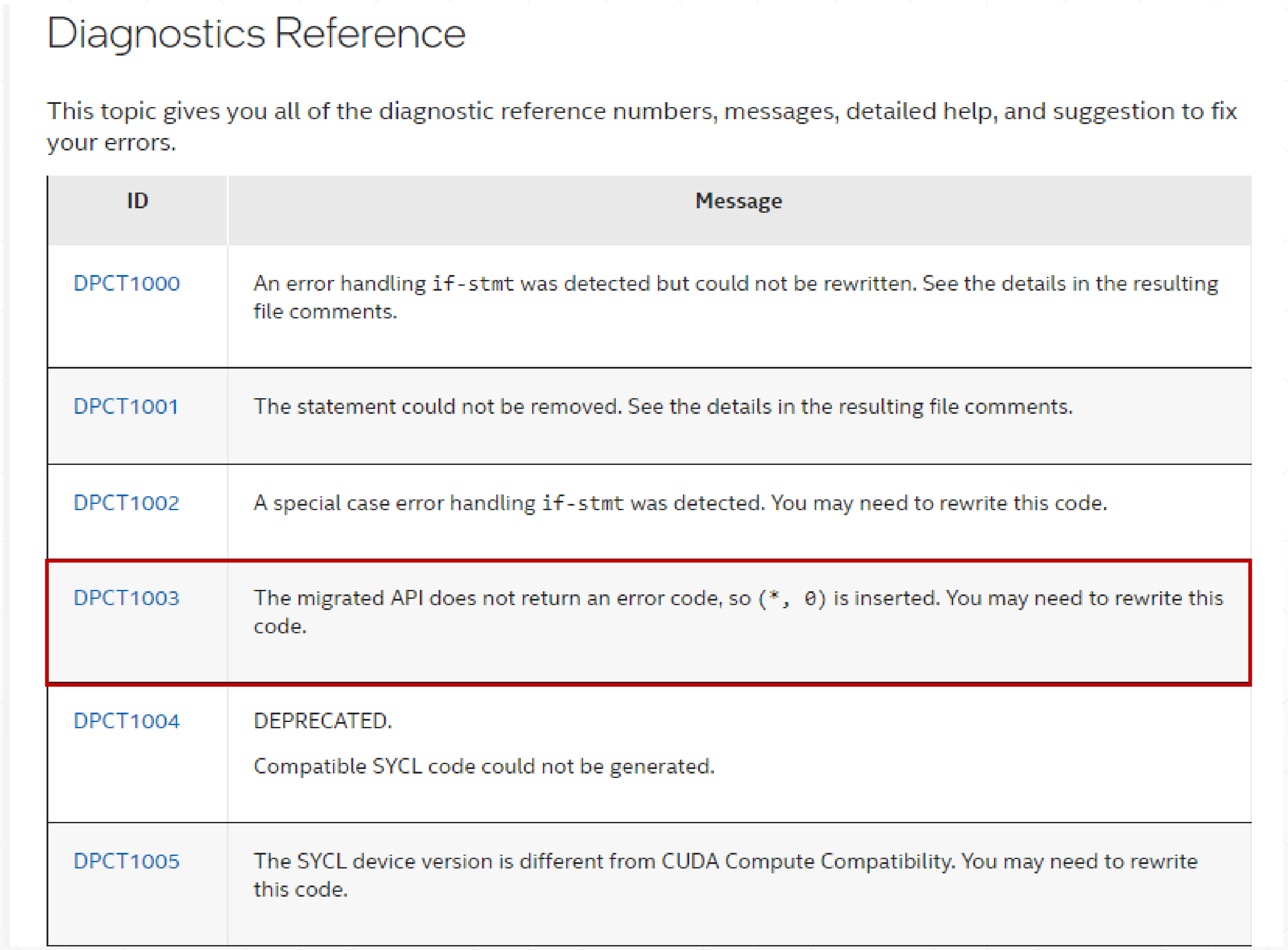
```
u66264@s019-n003:~$ dpct --cuda-include-path=/home/u66264/apps/build/cuda/include src/vector_add.cu
NOTE: Could not auto-detect compilation database for file 'vector_add.cu' in '/home/u66264/src' or any parent directory.
The directory "dpct_output" is used as "out-root"
Processing: /home/u66264/src/vector_add.cu
/home/u66264/src/vector_add.cu:32:14: warning: DPCT1003:0: Migrated API does not return error code. (*, 0) is inserted. You may need t
o rewrite this code.
    status = cudaMemcpy(Result, d_C, VECTOR_SIZE*sizeof(float), cudaMemcpyDeviceToHost);
             ^
Processed 1 file(s) in -in-root folder "/home/u66264/src"

See Diagnostics Reference to resolve warnings and complete the migration:
https://software.intel.com/content/www/us/en/develop/documentation/intel-dpcpp-compatibility-tool-user-guide/top/diagnostics-reference
.html
```



```
u66264@s019-n003:~$ ls -l dpct_output/
total 24
-rw------- 1 u66264 u66264  1991 Apr 10 22:28 vector_add.dp.cpp
-rw------- 1 u66264 u66264 17802 Apr 10 22:28 MainSourceFiles.yaml
```

- https://www.intel.com/content/www/us/en/docs/dpcpp-compatibility-tool/developer-guide-reference/2023-0/diagnostics-reference.html

SYCL Hands-on: Diagnostic Reference



Diagnostics Reference

This topic gives you all of the diagnostic reference numbers, messages, detailed help, and suggestion to fix your errors.

ID	Message
DPCT1000	An error handling `if-stmt` was detected but could not be rewritten. See the details in the resulting file comments.
DPCT1001	The statement could not be removed. See the details in the resulting file comments.
DPCT1002	A special case error handling `if-stmt` was detected. You may need to rewrite this code.
DPCT1003	The migrated API does not return an error code, so `(*, 0)` is inserted. You may need to rewrite this code.
DPCT1004	DEPRECATED. Compatible SYCL code could not be generated.
DPCT1005	The SYCL device version is different from CUDA Compute Compatibility. You may need to rewrite this code.

SYCL Hands-on: 커널 정의 (3-D ND Range)

CUDA

```
__global__ void VectorAddKernel(float* A, float* B, float* C)
{
    A[threadIdx.x] = threadIdx.x + 1.0f;
    B[threadIdx.x] = threadIdx.x + 1.0f;
    C[threadIdx.x] = A[threadIdx.x] + B[threadIdx.x];
}
```

DPC++

```
void VectorAddKernel(float* A, float* B, float* C, sycl::nd_item<3> item_ct1)
{
    A[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    B[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    C[item_ct1.get_local_id(2)] = A[item_ct1.get_local_id(2)]
                                + B[item_ct1.get_local_id(2)];
}
```

SYCL Hands-on: 메모리 할당 (USM)

CUDA

```
int main()
{
    float *d_A, *d_B, *d_C;
    cudaError_t status;

    cudaMalloc(&d_A, VECTOR_SIZE*sizeof(float));
    cudaMalloc(&d_B, VECTOR_SIZE*sizeof(float));
    cudaMalloc(&d_C, VECTOR_SIZE*sizeof(float));
```

DPC++

```
int main()
{
    dpct::device_ext &dev_ct1 = dpct::get_current_device();
    sycl::queue &q_ct1 = dev_ct1.default_queue();
    float *d_A, *d_B, *d_C;
    int status;

    d_A = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);
    d_B = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);
    d_C = sycl::malloc_device<float>(VECTOR_SIZE, q_ct1);
```

SYCL Hands-on: 커널 작동

CUDA

```
vectorAddKernel<<<1, VECTOR_SIZE>>>(d_A, d_B, d_C);

float Result[VECTOR_SIZE] = { };

status = cudaMemcpy(Result, d_C, SIZE*sizeof(float), cudaMemcpyDeviceToHost);
cudaMemcpyDeviceToHost);
```

DPC++

```
q_ct1.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, VECTOR_SIZE),
                                     sycl::range<3>(1, 1, VECTOR_SIZE)),
                                     [=](sycl::nd_item<3> item_ct1) {
                                       VectorAddKernel(d_A, d_B, d_C, item_ct1);
});

  float Result[VECTOR_SIZE] = { };

  /*
  DPCT1003:0: Migrated API does not return error code. (*, 0) is inserted. You
  may need to rewrite this code.
  */
  status = (q_ct1.memcpy(Result, d_C, VECTOR_SIZE * sizeof(float)).wait(), 0);
```

SYCL Hands-on: 메모리 해방

CUDA

```
  cudaFree(d_A);
  cudaFree(d_B);
  cudaFree(d_C);

  return 0;
}
```

DPC++

```
  sycl::free(d_A, q_ct1);
  sycl::free(d_B, q_ct1);
  sycl::free(d_C, q_ct1);

  retun 0;
}
```

SYCL Hands-on: 커널 정의 (1-D ND Range)

DPC++

```
void VectorAddKernel(float* A, float* B, float* C, sycl::nd_item<3> item_ct1)
{
    A[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    B[item_ct1.get_local_id(2)] = item_ct1.get_local_id(2) + 1.0f;
    C[item_ct1.get_local_id(2)] = A[item_ct1.get_local_id(2)]
                                  + B[item_ct1.get_local_id(2)];
}
```

$ dpct ***--assume-nd-range-dim=1*** *…*

DPC++

```
void VectorAddKernel(float* A, float* B, float* C, sycl::nd_item<3> item_ct1)
{
    A[item_ct1.get_local_id(0)] = item_ct1.get_local_id(0) + 1.0f;
    B[item_ct1.get_local_id(0)] = item_ct1.get_local_id(0) + 1.0f;
    C[item_ct1.get_local_id(0)] = A[item_ct1.get_local_id(0)]
                                  + B[item_ct1.get_local_id(0)];
}
```

SYCL Hands-on: 커널 작동 (C++17기준의 CTAD)

DPC++

```
q_ct1.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, VECTOR_SIZE),
                                     sycl::range<3>(1, 1, VECTOR_SIZE)),
                                     [=](sycl::nd_item<3> item_ct1) {
                                       VectorAddKernel(d_A, d_B, d_C, item_ct1);
});
```

$ dpct ***--enable-ctad*** *…*

- C++17 class template argument deduction

DPC++

```
q_ct1.parallel_for(sycl::nd_range(sycl::range(1, 1, VECTOR_SIZE),
                                  sycl::range(1, 1, VECTOR_SIZE)),
                                  [=](sycl::nd_item<3> item_ct1) {
                                     VectorAddKernel(d_A, d_B, d_C, item_ct1);
});
```