OpenMP Offload with Intel(R) oneAPI

oneAPI – 가속 컴퓨팅을 개발하기 위한 스마트한 방식

2023. 12. 18.

MOASYS

oneAPI 스마트한 방식 시리즈 (2023)

1. **Introducing Intel oneAPI 2023**
   - https://www.allshowtv.com/detail.html?idx=1259

2. **Heterogenous Programming with oneAPI and DPC++(SYCL)**
   - https://www.allshowtv.com/detail.html?idx=1337

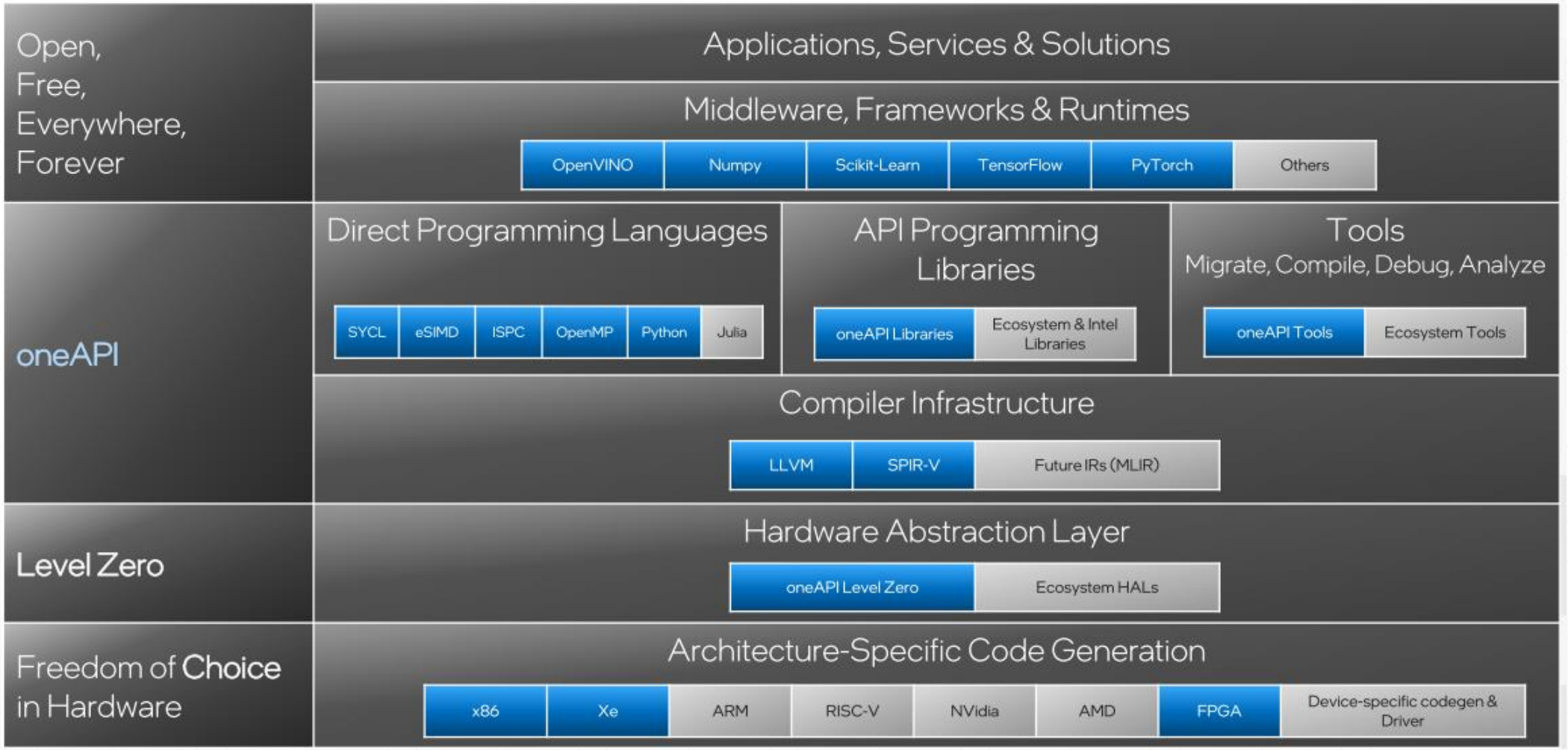
3. **CUDA to SYCL Migration with Intel(R) oneAPI**
   - https://www.allshowtv.com/detail.html?idx=1377

4. **OpenMP Offload with Intel(R) oneAPI**
   - https://www.allshowtv.com/detail.html?idx=1406

목차

- **Overview of oneAPI**
  - Intel compilers in oneAPI HPC Toolkit
  - Porting Guideline
- **OpenMP GPU Offload**
  - Introduction
  - Data management
  - GPU parallelism
  - Unified shared memory
  - oneMKL offload
- **Intel Developer Cloud**
- **Conclusion**
- **Handons**

아키텍처 간 가속 프로그래밍을 위한 스마트한 방식



- CPU, GPU 및 FPGA와 같은 다양한 가속기 (XPU) 지원
- 고성능 컴퓨팅 및 기계 학습을 위한 사양을 지속적으로 발전

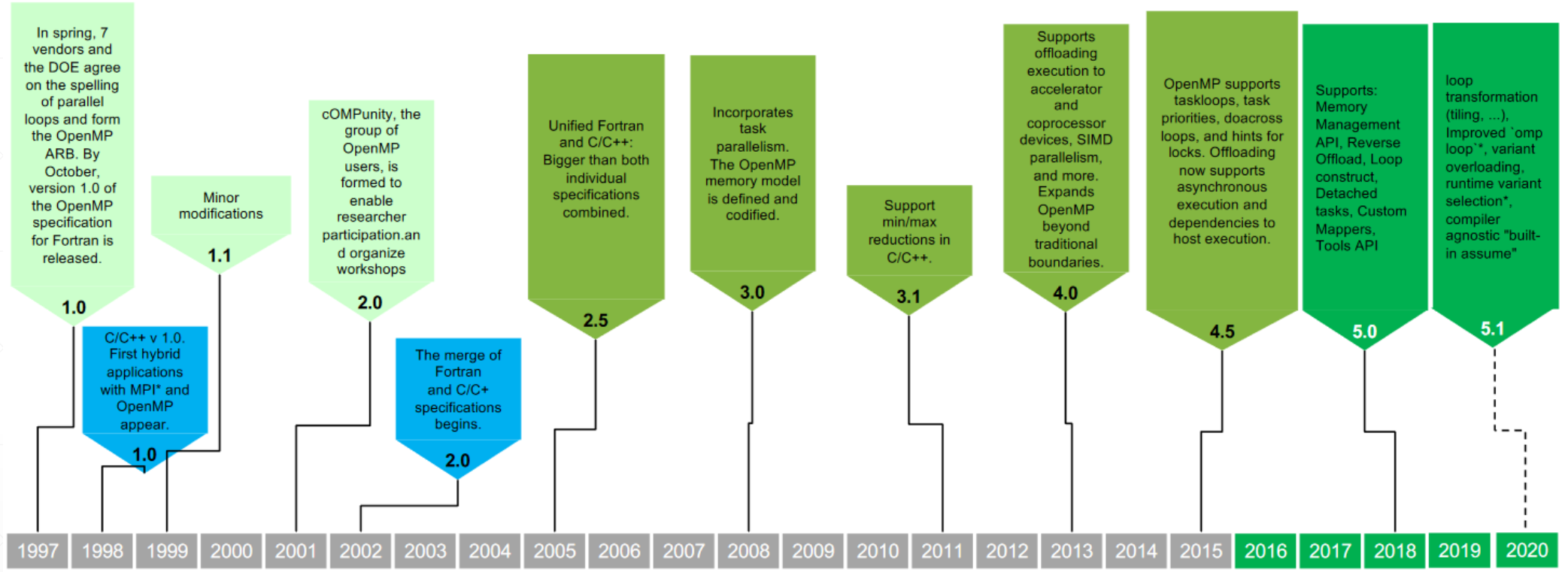
이기종 프로그래밍 모델 지원

	CUDA	HIP	OpenACC	OpenMP	DPC++/SYCL
Languages	C/C++/Fortran	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

OpenMP 표준의 연속적인 개발



- **OpenMP (Open Multi-Processing):**
  - C/C++/Fortran에서 공유 메모리 병렬 처리를 위한 오픈 소스, 이식성 및 디렉티브에 기반 API
  - OpenMP 4.0: 이기종 디바이스에 오프로딩 도입
  - OpenMP 4.5: 비동기된 실행 (nowait) 및 호스트의 실행에 의존 (depend) 도입
  - OpenMP 5.0: 디바이스 메모리 루틴 및 데이터 매핑 도입

Intel® Fortran 컴파일러 : Classic vs. Modern

- **ifort: Intel® Fortran Compiler Classic**
  - Intel(R) 옵티마이저 및 코드 생성
  - CPU만 지원함
  - 모든 Fortran 표준 지원가능함

- **ifx: Intel® Fortran Compiler**
  - Intel(R) 개선된 LLVM 옵티마이저 및 코드 생성
  - Intel(R) CPU/GPU로 오프로드 지원함
  - OpenMP 5.0 및 5.1의 주요 기능 지원함
  - Fortran의 주요 표준 지원함
    - Fortran 1995: 완전히 구현됨
    - Fortran 2003: 완전히 구현됨
    - Fortran 2008: 완전히 구현됨 (2022.2.0버전 부터 coarrays 포함)
    - Fortran 2018: 부분 구현

- **Reference:**
  - https://www.intel.com/content/www/us/en/developer/articles/technical/fortran-language-and-openmp-features-in-ifx.html

마이그레이션 안내

Compiler options	ifort	ifx
Disable optimization	-O0	-O0
Optimization for speed	-O1	-O1
Optimization for speed	-O2	-O2
High-level loop optimization	-O3	-O3
AVX512 vector width	-qopt-zmm-usage=high	**-mprefer-vector-width=512**
Floating point optimization	-fp model fast=2	-fp model fast=2 **-assume nan_compares**
Micoarchitecture optimization	-xSAPPHIRERAPIDS	-xSAPPHIRERAPIDS
Interprocedural optimization	-ipo	-ipo
OpenMP support	-qopenmp	-qopenmp or -fiopenmp
Just-in-time offload support	N/A	**-fopenmp-targets=spir64**
Ahead-of-time offload support	N/A	**-fopenmp-targets=spir64_gen -Xopenmp-target-backend "-device xehp"**
Optimization report	-qopt-report=5	**-qopt-report=3**
Optimization phase report	-qopt-report-phase=loop	N/A
Optimization report filter	-qopt-report-routine=stencile	N/A

- https://www.intel.com/content/www/us/en/developer/articles/guide/porting-guide-for-ifort-to-ifx.htm

OpenMP 오프로드: 핵심 아이디어

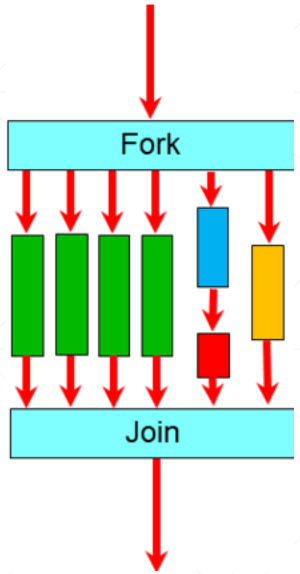
```
subroutine saxpy(a, x, y, n)
    real, intent(in)          :: a
    real, intent(in)          :: x(n)
    real, intent(inout)       :: y(n)
    integer, intent(in)       :: n
    integer                   :: i
    !$omp target teams distribute parallel do simd map(to: x) map(tofrom: y)
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
end subroutine
```

호스트와 디바이스 간에 데이터 전송

반복을 스레드에 배포하고 각 스레드는 SIMD 병렬 처리를 사용함

디바이스에서 병렬로 실행해야 하는 부분을 식별함

OpenMP 오프로드: target




```
subroutine saxpy(a, x, y, n)
    real, intent(in)          :: a
    real, intent(in)          :: x(n)
    real, intent(inout)       :: y(n)
    integer, intent(in)       :: n
    integer                   :: i
    !$omp parallel do
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end parallel do
end subroutine
```

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)          :: a
    real, intent(in)          :: x(n)
    real, intent(inout)       :: y(n)
    integer, intent(in)       :: n
    integer                   :: i
    !$omp target
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```

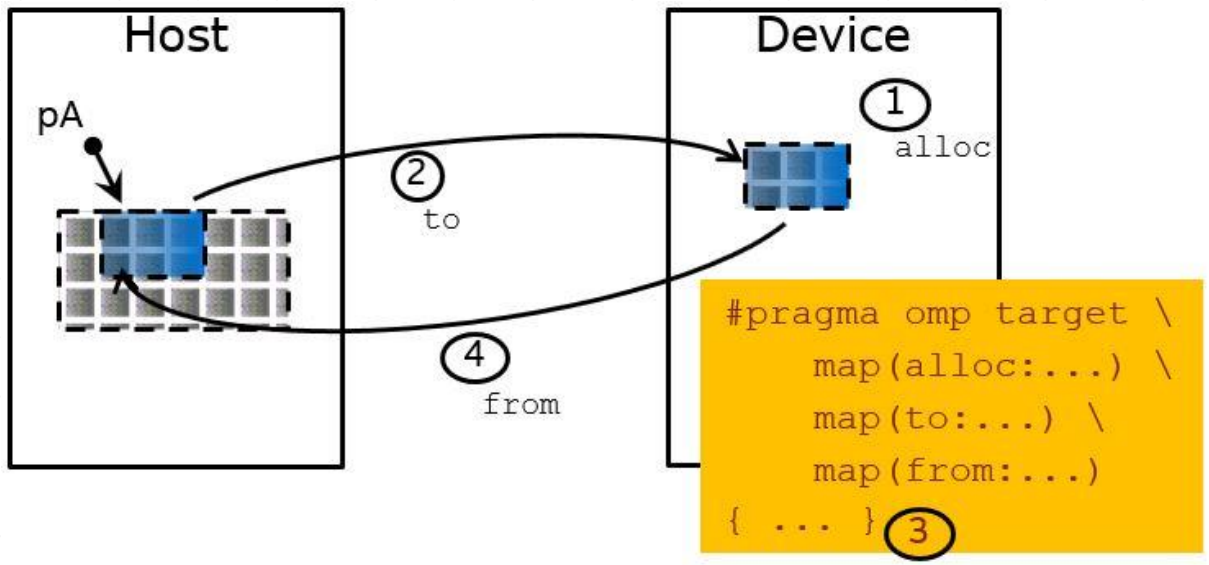
- target: 커널을 디바이스로 오프로드
  - C/C++:
    - *#pragma omp target [clause]*
  - Fortran:
    - *!$omp target [clause]*
      !$omp end target
  - clause:
    - *device(scalar-integer-expression)*
    - *if(scalar-expr)*
    - *map([{alloc | to | from | tofrom}:] list)*

OpenMP 오프로드: 암목적 데이터 이동

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)           :: a
    real, intent(in)           :: x(n)
    real, intent(inout)        :: y(n)
    integer, intent(in)        :: n
    integer                    :: i
    !$omp target
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```

host

device



- **OpenMP의 메모리 모델**
  - 호스트와 디바이스에는 별도의 메모리 공간 존재
  - 데이터를 액세스 위해 호스트에서 디바이스로 데이터를 이동해야 함

- **OpenMP의 암목적 이동**
  - Scalar (a): first private로 구성됨
  - Static arrays (x, y): tofrom로 구성됨
  - Dynamic arrays (x,y): tofrom로 구성됨

OpenMP 오프로드: 명시적 데이터 이동

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i


    !$omp target map(tofrom: y) map(to:x)
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```


```
subroutine saxpy_offload(a, x, y)
    real, intent(in)     :: a
    real, intent(in)     :: x(:)
    real, intent(inout)  :: y(:)
    integer              :: n
    integer              :: i

    n = size(x,1)
    !$omp target map(tofrom: y(1:n)) map(to:x(1:n))
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```

- **map:** 명시적 데이터 이동
  - alloc: 디바이스에 데이터를 할당하며 초기화하지 않음
  - to: target 진입 시 호스트에서 디바이스로 데이터 이동함
  - from: target 탈출 시 디바이스에서 호스트로 데이터 이동함
  - tofrom: to 및 from 같이 이용함

- C/C++의 포인터 및 Fortran의 dynamics/allocatable 경우에는 차원이 필요함

OpenMP 오프로드: 데이터 이동에 최적화 방식

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)      :: a
    real, intent(in)      :: x(n)
    real, intent(inout)   :: y(n)
    integer, intent(in)   :: n
    integer               :: i

    !$omp target map(from:x) map(fromto:y)
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target

    call init (y, n, 1.0)

    !$omp target map(from:x) map(fromto:y)
    do i=1, n
        y(i) = a * x(i) * y(i)
    end do
    !$omp end target
end subroutine
```

host

호스트에 데이터 할당

device

target 진입 시

- allocate(x, y) / host → device(x, y)

target 탈출 시

- device → host(y) / deallocate(x, y)

host

호스트에 y를 다시 초기화

device

target 진입 시

- allocate(x, y) / host → device(x, y)

target 탈출 시

- device → host(y) / deallocate(x, y)

- X 및 Y를 두 번 복사하여 중복된 데이터 이동됨

OpenMP 오프로드: target data

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)      :: a
    real, intent(in)      :: x(n)
    real, intent(inout)   :: y(n)
    integer, intent(in)   :: n
    integer               :: i

    !$omp target data map(to:x)
        !$omp target map (to:y)
        do i=1, n
            y(i) = a * x(i) + y(i)
        end do
        !$omp end target

        call init(y, n, 1.0)

        !$omp target map(tofrom:y)
        do i=1, n
            y(i) = a * x(i) * y(i)
        end do
        !$omp end target
    !$omp end target data
end subroutine
```

host

호스트에 데이터 할당

device

target 진입 시

- **allocate(x) / host -> device(x)**
- allocate(y) / host → device(y)

target 탈출 시

- device → host(y) / deallocate(y)

host

호스트에 y를 다시 초기화

device

target 진입 시

- allocate(y) / host → device(y)

target 탈출 시

- device → host(y) / deallocate(y)
- **deallocate(x)**

- x는 target data 통해 한 번 할당되고 복사됨

OpenMP 오프로드: target enter/target exit/target update

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)      :: a
    real, intent(in)      :: x(n)
    real, intent(inout)   :: y(n)
    integer, intent(in)   :: n
    integer               :: i

    !$omp target enter data map(to:x) map(to:y)
        !$omp target
        do i=1, n
            y(i) = a * x(i) + y(i)
        end do
        !$omp end target

        call init (y, n, 1.0)

        !$omp target update to(y)
        !$omp target
        do i=1, n
            y(i) = a * x(i) * y(i)
        end do
        !$omp end target
    !$omp target exit data map(from:y)
end subroutine
```

host

호스트에 데이터 할당

device

target 진입 시

- allocate(x, y) / host → device (x, y)

host

호스트에 y를 다시 초기화

device

디바이스의 y를 update

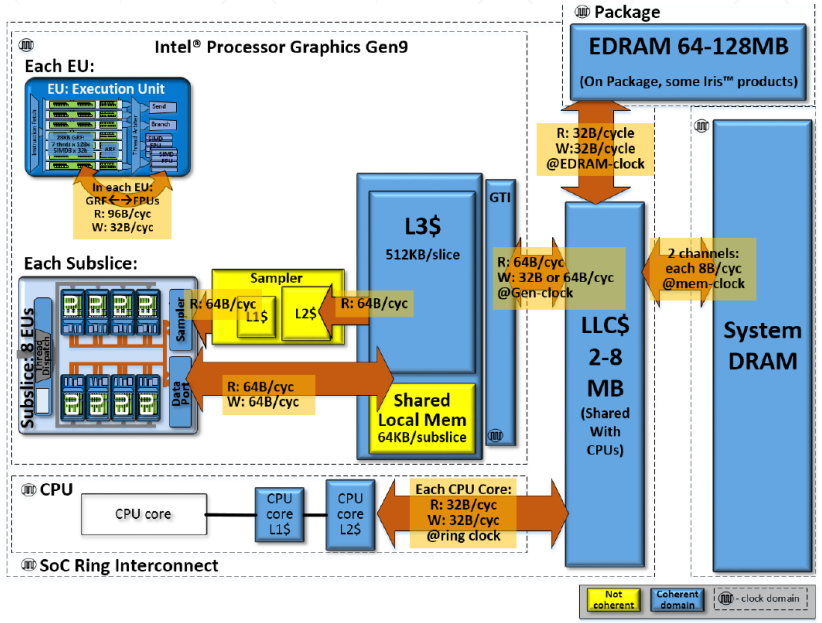
- host → device(y)

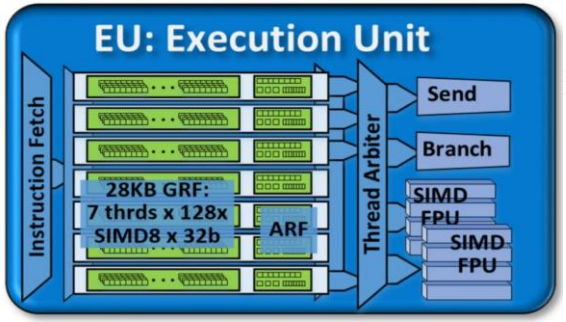
target 탈출 시

- device → host(y) / deallocate(x, y)

OpenMP 오프로드: 디바이스 병령화

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i
    !$omp target map(tofrom: y) map(to:x)
    !$omp parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```





- 1 x slice
- 3 x subslices
- 8 x EUs per subslice
- 7 x threads per EU
- 128 x SIMD8 lanes per thread

- **OpenMP 오프로드의 실행 모델**
  - 오프로드 및 병렬화 처리의 분리
  - target은 단일 디바이스 스레드를 생성합니다.

- **명시적 병령화 필요함**
  - *parallel do simd* 는 OpenMP 스레드를 단일 GPU의 sub-slice에 매핑합니다.
    - Sub-slice 간 동기화 없음

OpenMP 오프로드: teams

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i
    !$omp target map(tofrom: y) map(to:x)
    !$omp teams distribute parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```

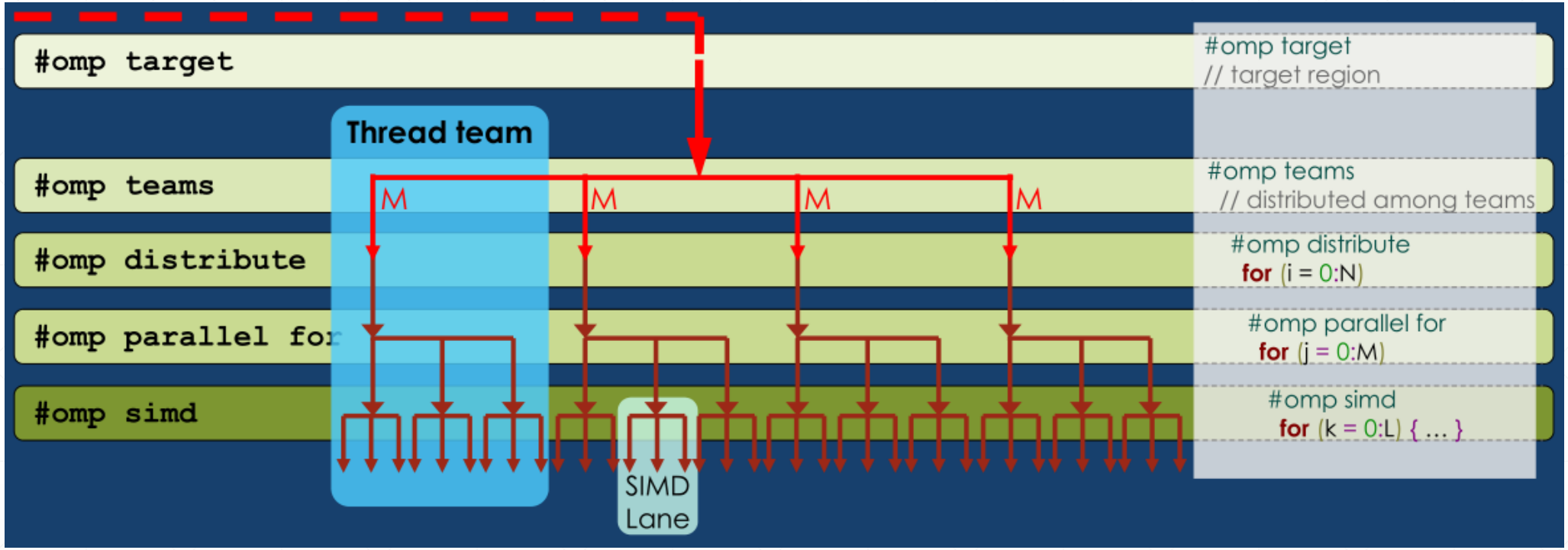
- **teams: sub-slice에 매핑된 master 스레드 생성**
  - **C/C++:**
    - *#pragma omp team [clause]*
  - **Fortran:**
    - *!$omp team [clause]*
  - **clause:**
    - *num_teams(integer-expression), thread_limit(integer-espression)*
    - *default(shared | first private | private none)*
    - *private(list), firstprivate(list), shared(list), reduction(operation:list)*

OpenMP 오프로드: distribute

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i
    !$omp target map(tofrom: y) map(to:x)
    !$omp teams distribute parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```

- **distribute: 여러 팀에 걸쳐 루프 반복을 배포**
  - **C/C++:**
    - *#pragma omp distribute [clause]*
  - **Fortran:**
    - *!$omp distribute [clause]*
  - **clause:**
    - *collapse(n)*
    - *dist_schedule(static[,chunk_size])*
    - *Firstprivate(list), lastprivate(list), private(list)*

OpenMP 오프로드: 디바이스 병령화



	NVIDIA offload	Intel offload
!$omp target	Device	Device
!$omp teams	SM	Subslice
!$omp parallel do	Warp	EU
!$omp simd	Thread	SIMD Lanes

OpenMP 오프로드: 호스트 디바이스 동시성

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i

    !$omp target map(tofrom: y) map(to:x)
    !$omp teams distribute parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target


    call host_work()


    print *, y(0)
end subroutine
```

host

device

host

```
subroutine saxpy_offload(a, x, y, n)
    real, intent(in)     :: a
    real, intent(in)     :: x(n)
    real, intent(inout)  :: y(n)
    integer, intent(in)  :: n
    integer              :: i

    !$omp target map(tofrom: y) map(to:x) nowait
    !$omp teams distribute parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target


    call host_work()

    !$omp taskwait
    print *, y(0)
end subroutine
```

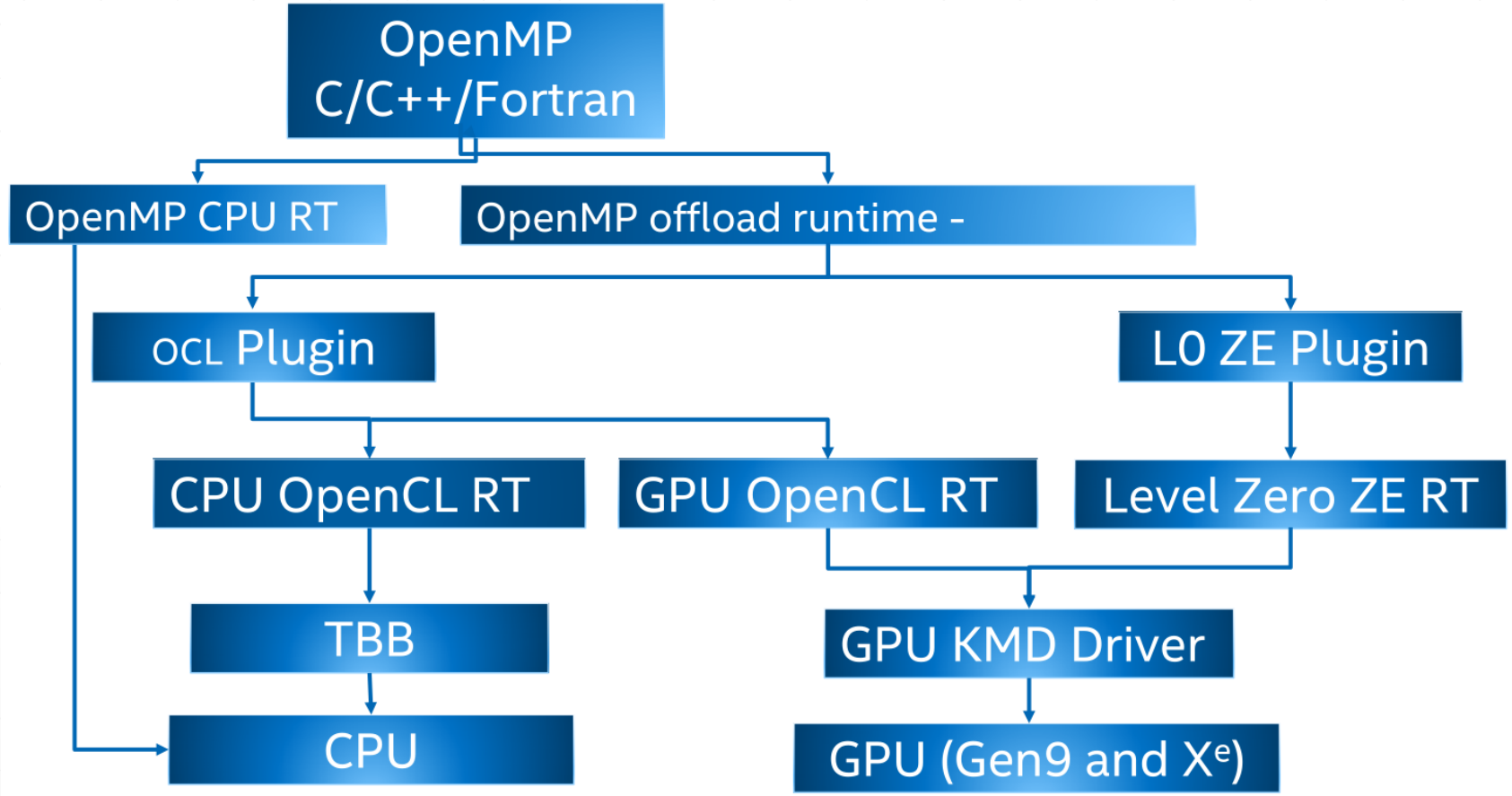
host

device

host

- **nowait**: 비동기 오프로딩 활성화함
- **taskwait**: 호스트 다시 동기화함

oneAPI의 OpenMP 오프로드 아키텍처



- **oneAPI BASE Toolkit**
  - OpenCL 백엔드: Intel(R) CPU 및 GPU에 최적화
  - Level Zero 백엔드: Intel(R) GPU를 위한 low level 오프로드 API

OpenMP 환경 변수

- 실행시 디바이스 선택
  - `OMP_TARGET_OFFLOAD = mandatory | disable | default`
    - mandatory: GPU로 오프로드
    - disabled: CPU로 오프로드
    - default: 가능한 경우에는 GPU로 오프로드 하고 그렇지 않으면 CPU로 이용함
- OpenMP의 백엔드 선택
  - `LIBOMPTARGET_PLUGIN = opencl | level0`
- Performance profiling on GPU
  - `LIBOMPTARGET_PLUGIN_PROFILE = T | F`

```
LIBOMPTARGET_PLUGIN_PROFILE(LEVEL0) for OMP DEVICE(0) Intel(R) Graphics [0x020a], Thread 0
-------------------------------------------------------------------------------------------------------
Kernel 1                  : __omp_offloading_3b_dd7d2220_MAIN___l36
-------------------------------------------------------------------------------------------------------
: Host Time (msec)                    Device Time (msec)
Name                      :   Total  Average      Min      Max    Total  Average      Min      Max  Count
-------------------------------------------------------------------------------------------------------
Compiling                 : 2586.27  2586.27  2586.27  2586.27     0.00     0.00     0.00     0.00   1.00
DataAlloc                 :    5.93     0.42     0.00     1.29     0.00     0.00     0.00     0.00  14.00
DataRead (Device to Host) :   11.70    11.70    11.70    11.70     2.39     2.39     2.39     2.39   1.00
DataWrite (Host to Device):   29.86     3.32     0.09     9.72     8.23     0.91     0.00     2.74   9.00
Kernel 1                  : 7660.45  7660.45  7660.45  7660.45  7638.77  7638.77  7638.77  7638.77   1.00
Linking                   :    0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00   1.00
OffloadEntriesInit        :  807.01   807.01   807.01   807.01     0.00     0.00     0.00     0.00   1.00
```

DPC++ 및 OpenMP Interoperability

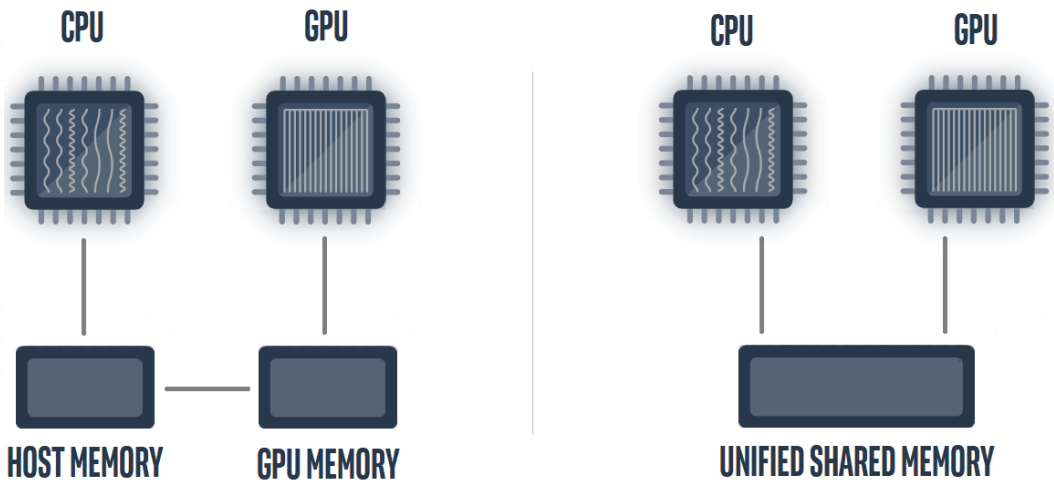
```
#include <CL/sycl.hpp>
#include <array>
#include <iostream>

float computePi(int N) {
    float Pi;
    #pragma omp target map(from : Pi)
    #pragma omp parallel for reduction(+ : Pi)
    for (unsigned I = 0; I < N; ++I) {
        float T = (I + 0.5f) / N;
        Pi += 4.0f / (1.0 + T * T);
    }
return Pi / N;
}

void iota(float *A, int N) {
    range<1> R(N);
    buffer<int,1> X(A, R);
    queue().submit([&](handler &cgh) {
        auto Y = X.template get_access<access::mode::write>(cgh);
        cgh.parallel_for<class Iota>(R, [=](id<1> idx) {
            Y[idx] = idx;
        });
    });
}
```

```
float computePi(int);
void iota(float*, int);

int main() {
    std::array<int, 1024> V;
    float Pi;

     #pragma omp parallel sections
     {
         #pragma omp section
         iota(V.data(), V.size());

         #pragma omp section
         Pi = computePi(8192);
      }

      return 0;
}
```

```
$ icpx \
      -fiopenmp \
      -fopenmp-targets=spir64 \
      -fsycl \
      -O2 \
      mixed.cpp
```

Unified Share Memory (USM)

```
program main
    use omp_lib
    integer, parameter           :: n = 1024
    real, allocatable            :: x(:), y(:)
    !$omp allocate allocator(omp_target_shared_mem_alloc)
    allocate(x(n))

    allocate(y(n))
    …
end program

subroutine saxpy_offload(a, x, y)
    real, intent(in)    :: a
    real, intent(in)    :: x(:)
    real, intent(inout) :: y(:)
    integer             :: n, i
    n = size(y,1)
    !$omp target map(tofrom: y(1:n)) has_device_addr(x)
    !$omp teams distribute parallel do simd
    do i=1, n
        y(i) = a * x(i) + y(i)
    end do
    !$omp end target
end subroutine
```



Type	Location	Accessible From	allocate directive
Host	Host	Host or Device	!$omp allocate allocator(omp_target_host_mem_alloc)
Device	Device	Device	!$omp allocate allocator(omp_target_device_mem_alloc)
Shared	Host or Device	Host or Device	!$omp allocate allocator(omp_target_shared_mem_alloc)

- has_device_addr(list)
  - Items has a valid memory address on device
  - Can be any types, including arrays

oneMKL: GPU Offload

```
include "common_blas.f90"
program dgemm
    use onemkl_blas_omp_offload_lp64
    use common_blas

    integer                          :: m = 1024, n = 1024, k = 256
    double precision                 :: alpha = 1.0, beta = 1.0
    double precision, allocatable    :: a(:,:), b(:,:), c_gpu(:,:), c_cpu(:,:)

    ! Array initialization
    call dinit_matrix('N', m, k, m, a)
    ...

    ! Call DGEMM on CPU for reference
    call dgemm('N', 'N', m, n, k, alpha, a, m, b, k, beta, c_cpu, m)

    ! Map matrices to device memory
    !$omp target data map(a,b,c_gpu)
        ! Call DGEMM on GPU
        !$omp target variant dispatch use_device_addr(a,b,c_gpu)
        call dgemm('N', 'N', m, n, k, alpha, a, m, b, k, beta, c_gpu, m)
        !$omp end target variant dispatch
    !$omp end target data

    ! free memory
    deallocate(a)
    ...
end program
```

host

device

host

호스트에 데이터 할당

호스트에서 초기화

호스트에서 matmul 실행

target 진입 시

- host → device (a,b,c_gpu)
- **use_device_addr():**
  - 호스트에서 array 액세스 가능.
- **target variant dispatch():**
  - dgemm의 GPU 버전 선택

호스트에서 데이터 할당 해제

OneAPI 구매 관련

- Purchasing oneAPI software with support provides registered users with many benefits shown on this page https://www.intel.com/content/www/us/en/developer/tools/oneapi/commercial-base-hpc.html and listed here:
  - Direct and private interaction with Intel's support engineers, including the ability to submit confidential support requests
  - Accelerated response time for technical questions and other product needs
  - Free download access to all new product updates and continued access to older versions of the product
  - Priority assistance for escalated defects and feature requests
  - Access to a vast library of self-help documentation built from decades of experience with creating high-performance code

- oneAPI
  - moasys 홈페이지에서 oneAPI 관련 workshop에 대한 정보를 확인하실 수 있습니다.
  - moasys 홈페이지 : http://www.moasys.com/
  - oneAPI 관련 정보 : http://www.moasys.com/oneapi.html

- oneAPI 구매 문의 : sales@moasys.com

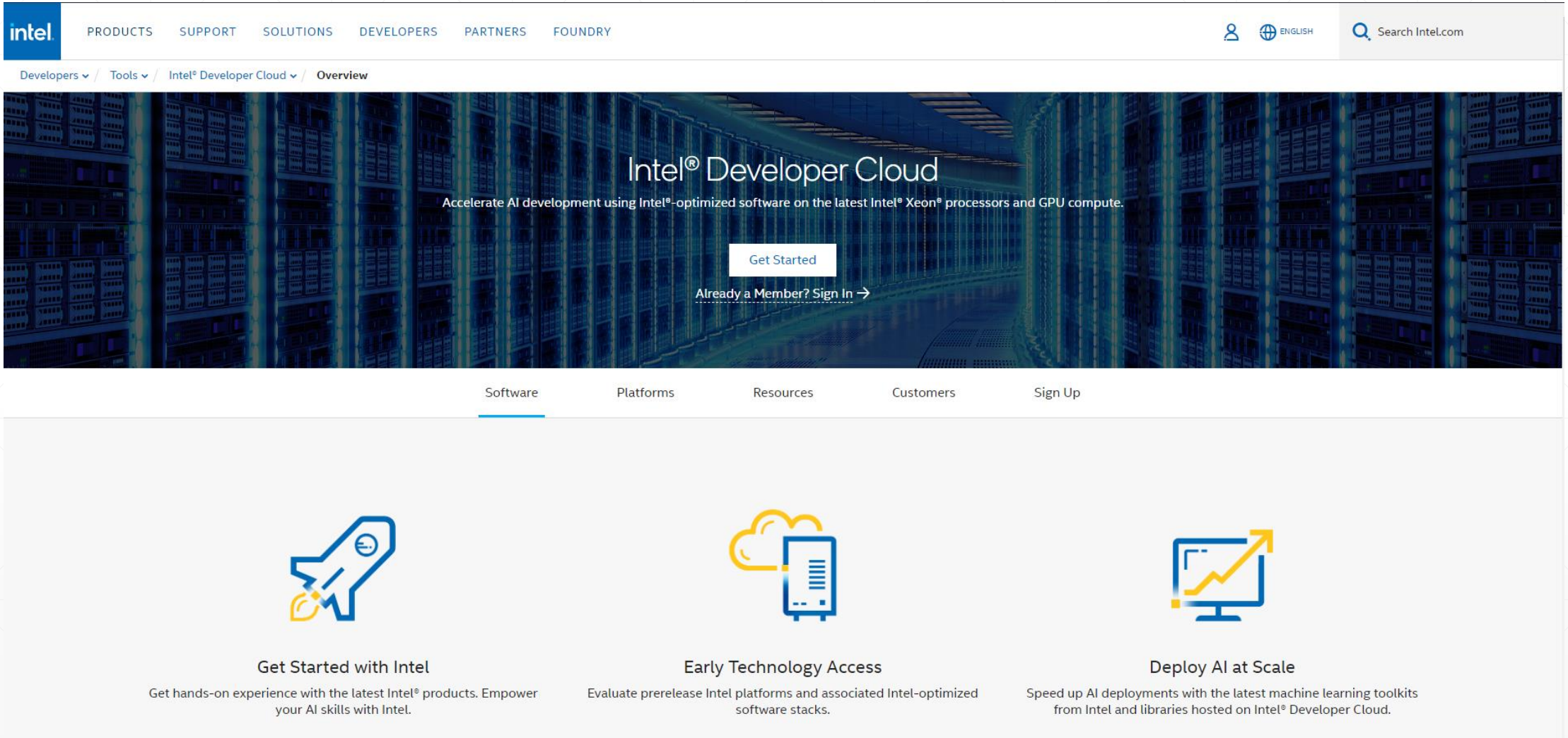
Intel(R) Developer Cloud Service

- **CPU platforms with VM/bare-metal access**
  - 4th Gen Intel(R) Xeon(R) Scalable Processors
  - Intel(R) CPU Max Series Processors
  - Coming soon: pre-production 5th Intel(R) Xeon(R)
- **GPU platforms with bare-metal access**
  - Intel(R) Data Center GPU Max Series
  - Intel(R) Data Center GPU Flex Series
- **AI accelerator platform with bare-metal access**
  - Hanaba Gaudi2(R) Processor for deep learning
  - Available for pre-selected qualified customers
- **Optimized software and tools**
  - Intel(R) oneAPI Base/HPC Toolkit
  - Intel(R) AI tools and optimized toolkits: Tensorflow, Pytorch2
  - Intel(R) Quantum SDK
- **AI Services:**
  - Stable Diffusion
  - Llama2
  - etc

Intel(R) Developer Cloud Service

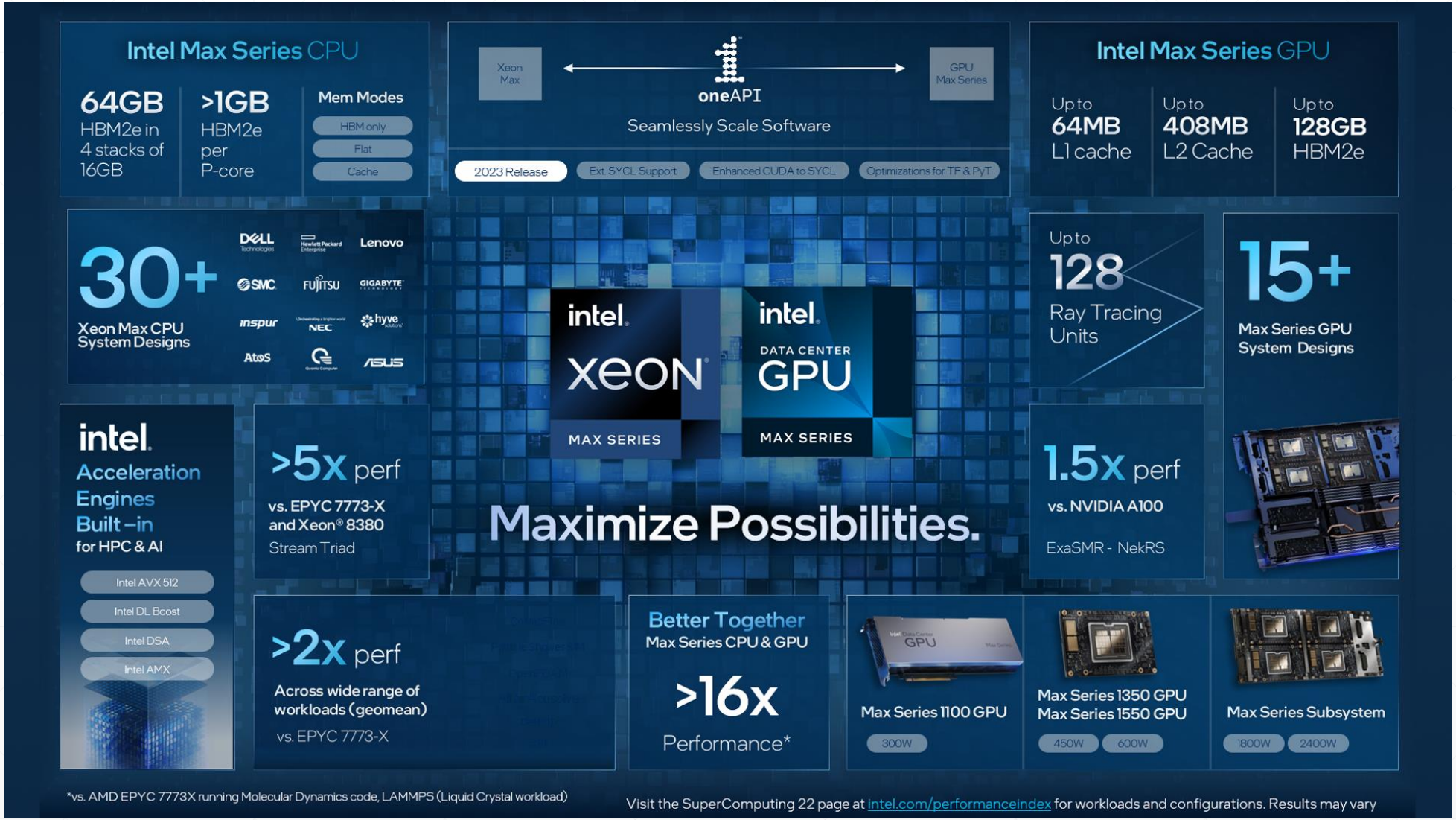
	Standard	Premium	Enterprise
Users	AI & HPC developers, data scientists, researchers, academia Single-user access	Enterprise customers/developers Single-user access	Enterprise customers Multi-user access
Usages	• Evaluation—test applications, workloads, & LLM workload code samples on different architectures • Build AI & HPC applications, optimize for new features & best performance • Schedule GPU access • Training & development, obtain Intel® Certified Developer accreditation for AI development & design	• Standard services usages + • AI training & inference production workloads • Model optimization & deployment • Certification, software validation & benchmark testing	• Premium services + • High-performance, cost-optimized Intel compute services for third-party AI SaaS providers
Hardware access	• 4th Gen Intel® Xeon® Scalable processors—bare metal & virtual machine (VM) • Intel® Xeon® CPU Max Series processors—bare metal • Intel® Data Center GPU Max Series & Intel® Data Center GPU Flex Series—bare metal • Habana® Gaudi®2 processors for Deep Learning—bare metal access for pre-qualified, select customers	• Standard services access + • Intel® Xeon® Scalable processors & Intel Data Center GPU Max & Flex Series—bare metal access to single node systems & clusters • Access to k8s clusters	Same as Premium services
Technical support	Community forum support (no SLA)	Support through Intel technical engineers Monday-Friday, 8 a.m. to 5 p.m. (per user's local region) 1 business day SLA	Premium Support through Intel technical engineers (phone, chat, help request tickets) 1 hour to 1 business day SLA, 24x7
Cost	**Free** + available with cloud credits for certain instance types such as bare metal services Optional upgrade option for extended use, pay-as-you-go	Available with cloud credits + based on an hourly rate noted in Intel Developer Cloud portal Discounts for long-term contracts/ reserve pricing are available, contact your Intel representative	Available with cloud credits + monthly subscription rate noted in Intel Developer Cloud porta. Discounted founders rate & long-term contracts/reserve pricing are available, contact your Intel representative

Intel(R) Developer Cloud Service



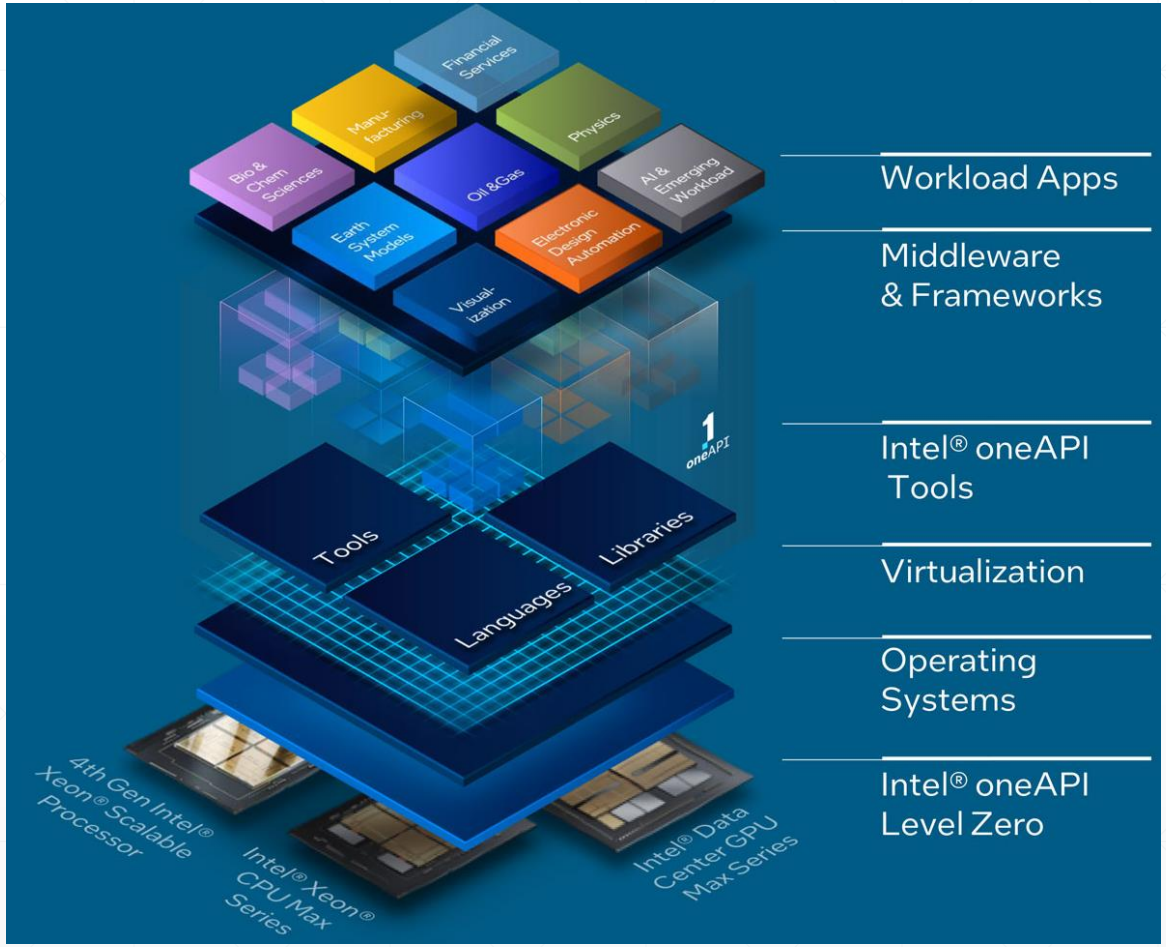
https://www.intel.com/content/www/us/en/developer/tools/devcloud/services.html

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
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
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 Xe 코어 / 64GB HBM / 300W
    - 인텔 Xe 링크 브릿지를 통해 여러 카드 연결 가능
  - Intel® 맥스 시리즈 1350 GPU
    - 112개 Xe 코어 / 96GB HBM / 450W
  - Intel® 맥스 시리즈 1550 GPU
    - 128개 Xe 코어 / 128GB HBM / 600W
- 개방형 표준 기반 교차 아키텍처 프로그래밍 프레임워크인 oneAPI로 통합이 가능



Intel® 맥스 시리즈 성능

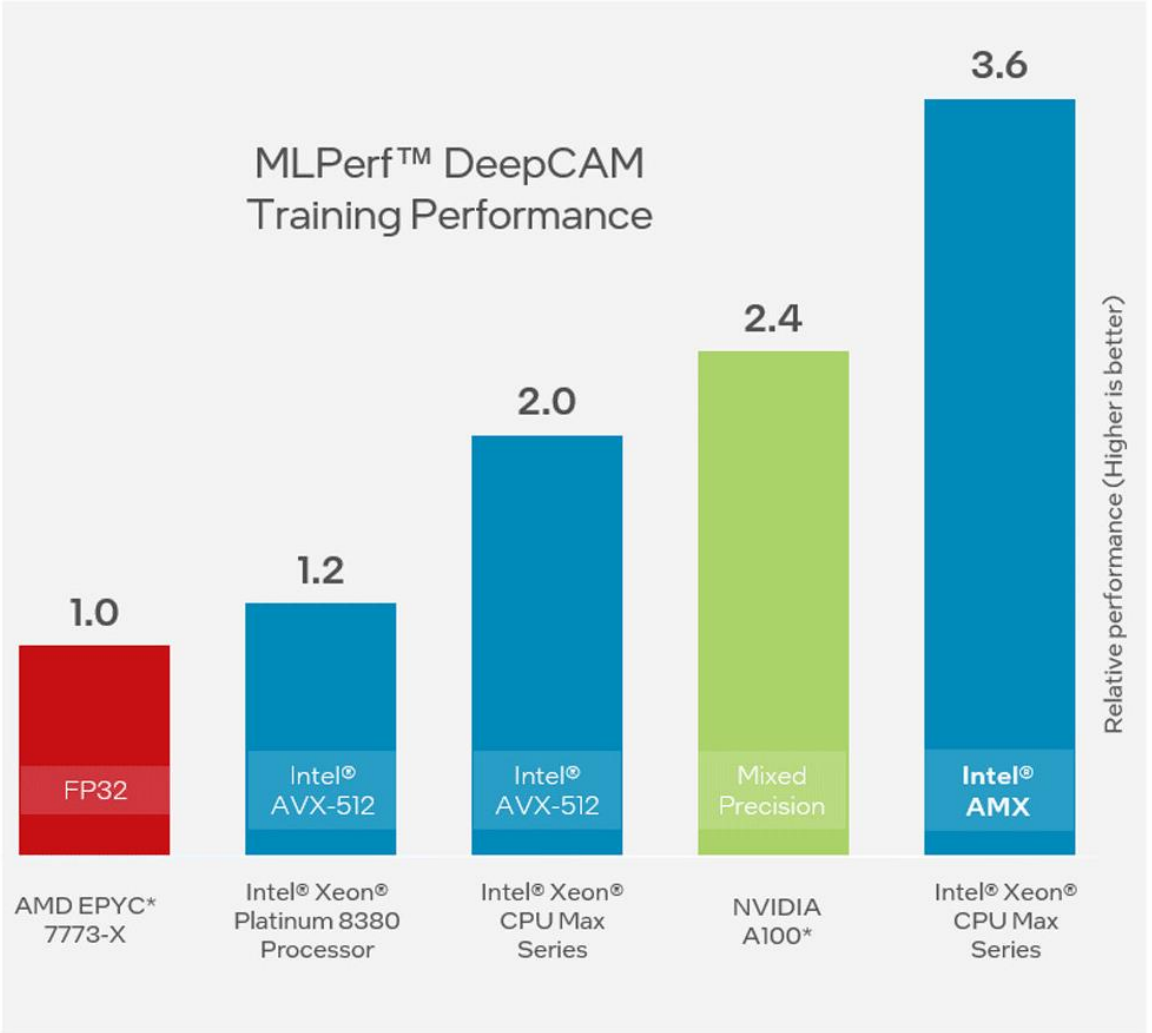
intel

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

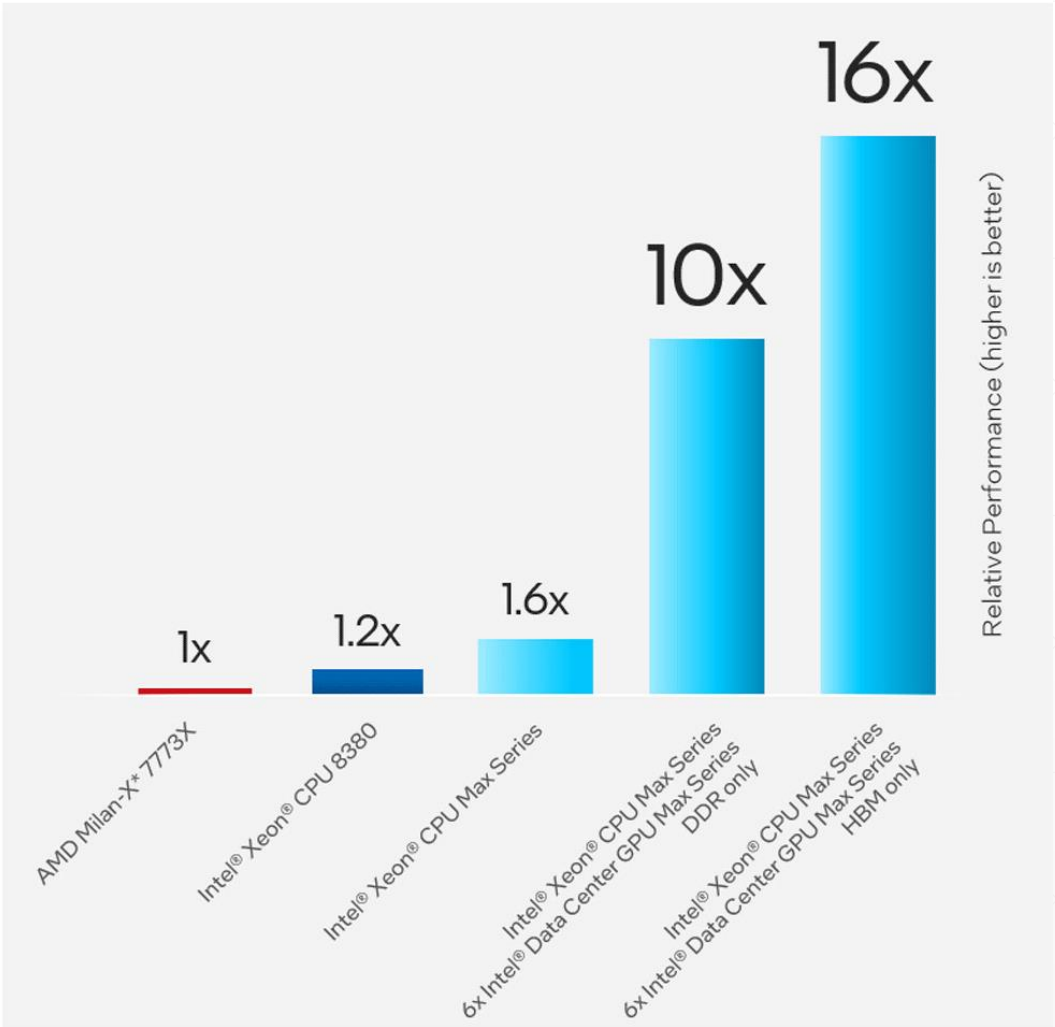
Speeding Exascale Material Discovery with Compute Density and HBM for LAMMPS (Liquid Crystal)

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

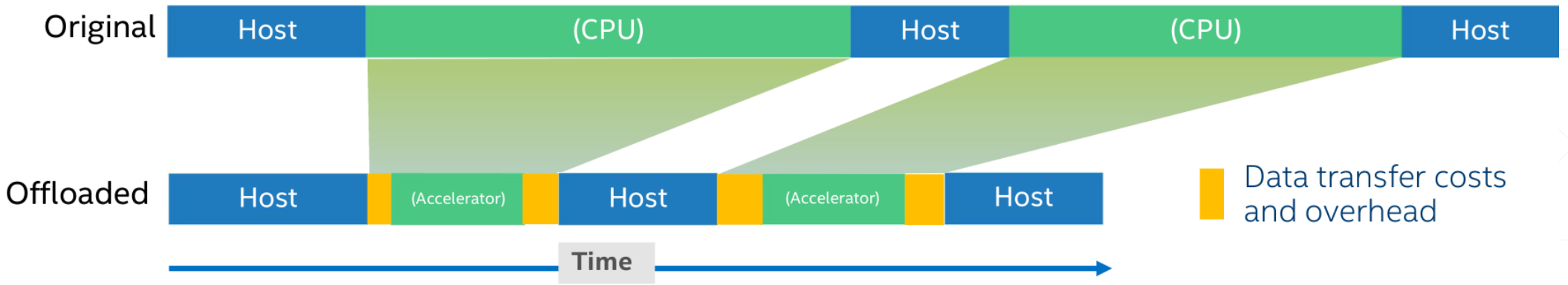
- Intel® Xeon® CPU Max Series: 1-node, 2x Intel® Xeon® CPU Max Series, HT On, Turbo On, Total Memory 128 GB HBM2e, BIOS EGSDCRB1.DWR.0085.D12.2207281916, ucode 0xac000040, SUSE Linux Enterprise Server 15 SP3, Kernel 5.3.18, oneAPI 2022.3.0, LAMMPS built with the Intel package
- Intel® Data Center GPU 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.

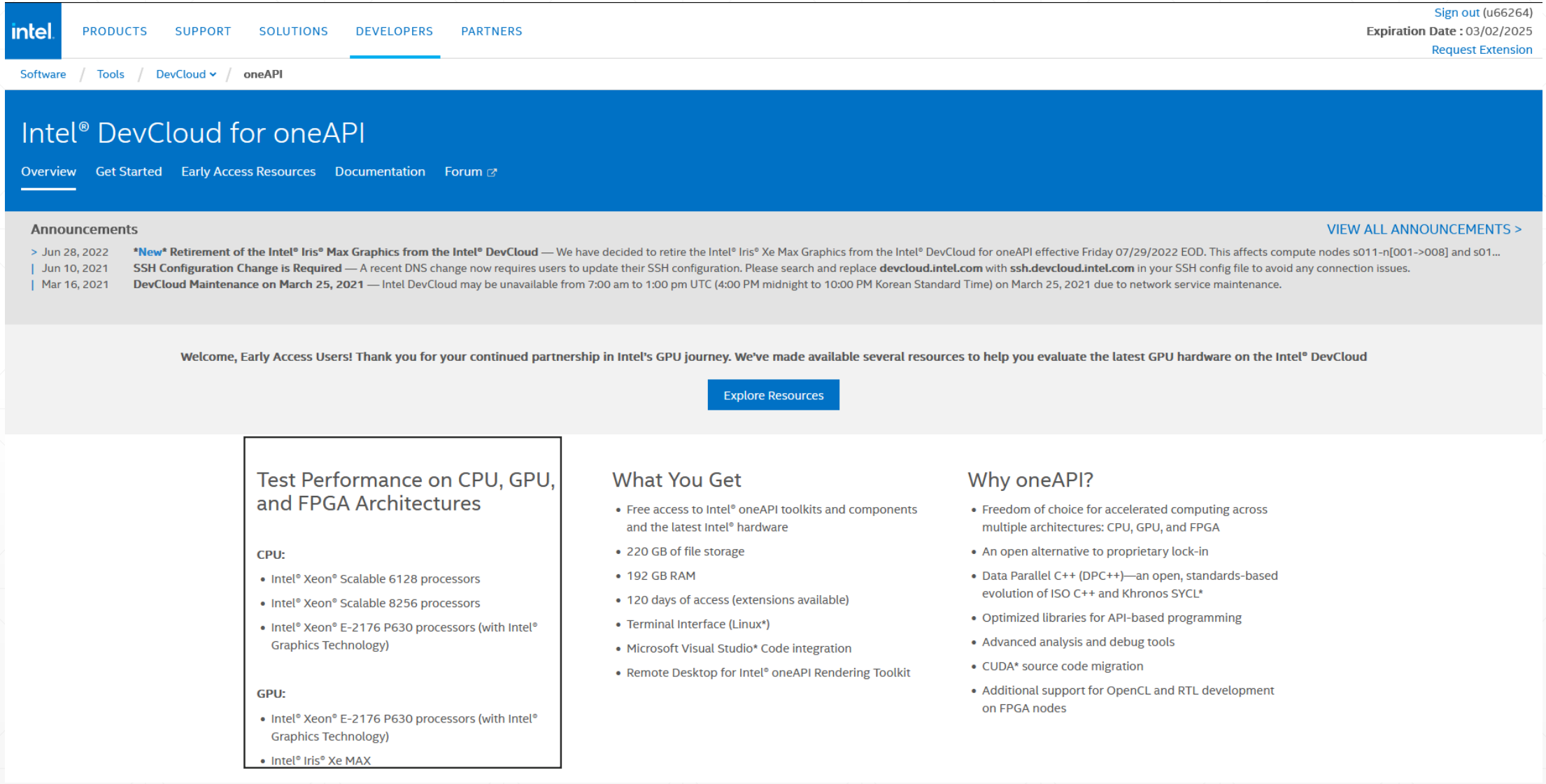


결론

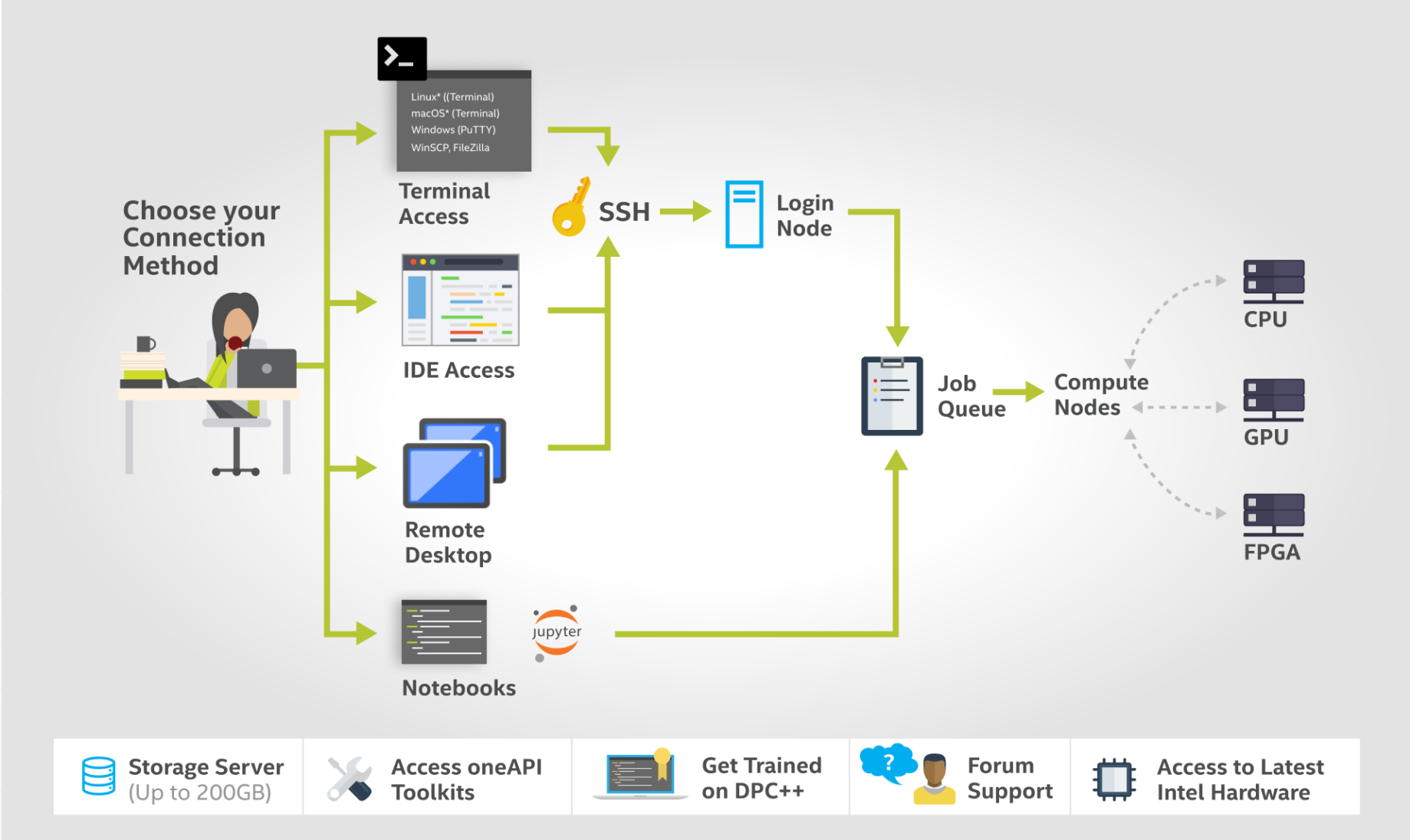


- New Intel® Fortran/C/C++ compilers in oneAPI
  - Classic Intel® compilers will enter legacy product support (icc: 2023, ifort: 2024)
  - Migration toward new Intel® compilers are recommended
- OpenMP offload
  - Data management
  - GPU parallelism
  - Unified share memory (USM)
  - oneMKL offload
- Contact: sales@moasys.com
  - GPU, FPGA porting
  - Optimization and parallelization consultant
  - Specialized HPC education

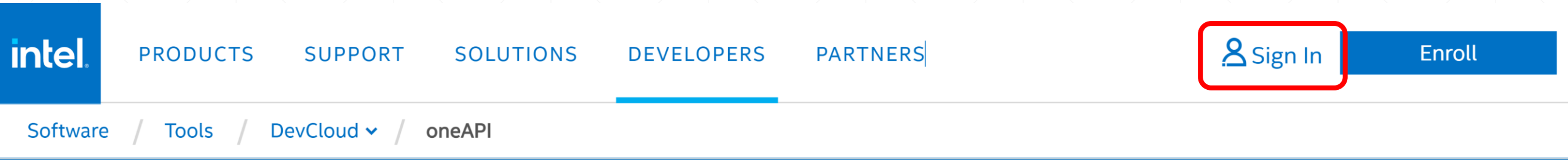
Intel(R) DevCloud for oneAPI



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

Overview Get Started Documentation Forum

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.

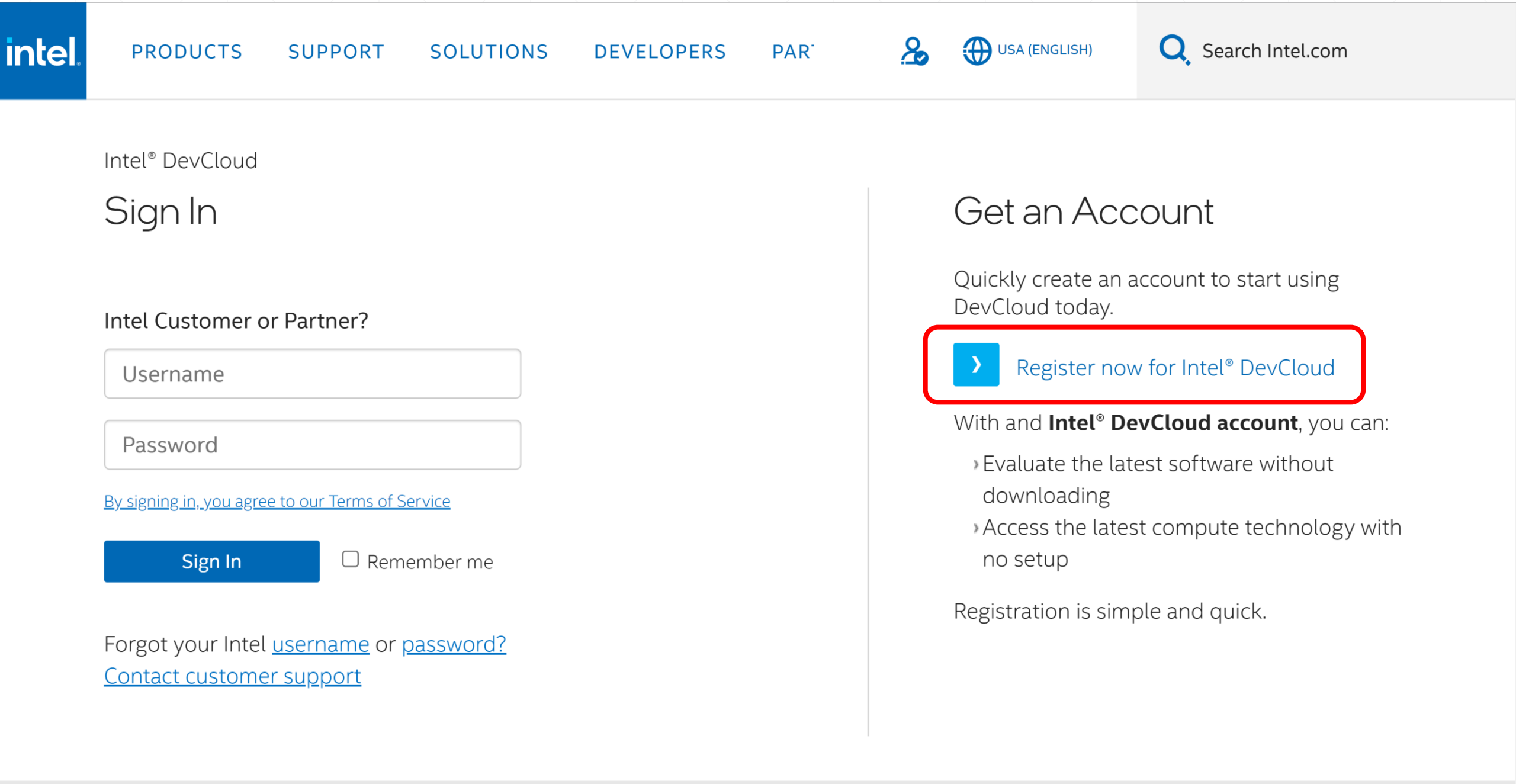
Get Free Access

Sign in

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



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.

Interactive Job

```
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 / iris_xe_max / aria10 / stratix10 / gold6128 / i9-10920x
- Properties describing purpose:
  - fpga_compile / fpga_runtime / renderkit

Ex1: Basic Query

```
program hello
  use omp_lib
  implicit none

  integer :: num_devices,nteams,nthreads
  logical :: initial_device

  num_devices = omp_get_num_devices()
  print *, "Number of available devices", num_devices

  !$omp target map(nteams,nthreads)
    initial_device = omp_is_initial_device()
    nteams= omp_get_num_teams()
    nthreads= omp_get_num_threads()
  !$omp end target

  if (initial_device) then
     write(*,*) "Running on host"
  else
     write(*,'(A,I4,A,I4,A)') "Running on device with ",nteams, " teams in total and ", nthreads, " threads in each team"
  end if

end program
```

- Compile OpenMP offload code

```
$ ifx -qopenmp -fopenmp-targets=spir64 query.90 –o query.x
```

Ex1: Basic Query (gen9)

- Offload mode

```
$ OMP_TARGET_OFFLOAD=mandatory ./query.x

Number of available devices           1
Running on device with    1 teams in total and    1 threads in each team
```

- Debug information
  - Level Zero selected by default (GPU only)

```
$ OMP_TARGET_OFFLOAD=mandatory LIBOMPTARGET_DEBUG=1 ./query.x

Libomptarget --> Init target library!
Libomptarget --> Callback to __tgt_register_ptask_services with handlers 0x00007f5b5bb10e40
0x00007f5b5bb11940
Libomptarget --> Initialized OMPT
Libomptarget --> Loading RTLs...
Libomptarget --> Loading library 'libomptarget.rtl.level0.so'...
Target LEVEL0 RTL --> Init Level0 plugin!
Libomptarget --> Successfully loaded library 'libomptarget.rtl.level0.so'!
```

Ex1: Basic Query (gen9)

- Debug information (cont)
  - Level Zero backend selected by default (GPU only)

```
$ OMP_TARGET_OFFLOAD=mandatory LIBOMPTARGET_DEBUG=1 ./query.x
...
Target LEVEL0 RTL --> Looking for Level0 devices...
Target LEVEL0 RTL --> Found a GPU device, Name = Intel(R) UHD Graphics
Target LEVEL0 RTL --> Found 1 root devices, 1 total devices.
Target LEVEL0 RTL --> List of devices (DeviceID[.SubID[.CCSID]])
Target LEVEL0 RTL --> -- 0
Target LEVEL0 RTL --> Root Device Information
Target LEVEL0 RTL --> Device 0
Target LEVEL0 RTL --> -- Name                          : Intel(R) UHD Graphics
Target LEVEL0 RTL --> -- PCI ID                        : 0x9a60
Target LEVEL0 RTL --> -- Number of total EUs           : 32
Target LEVEL0 RTL --> -- Number of threads per EU      : 7
Target LEVEL0 RTL --> -- EU SIMD width                 : 8
Target LEVEL0 RTL --> -- Number of EUs per subslice    : 16
Target LEVEL0 RTL --> -- Number of subslices per slice: 2
Target LEVEL0 RTL --> -- Number of slices              : 1
Target LEVEL0 RTL --> -- Local memory size (bytes)     : 65536
Target LEVEL0 RTL --> -- Global memory size (bytes)    : 26545831936
Target LEVEL0 RTL --> -- Cache size (bytes)            : 1966080
Target LEVEL0 RTL --> -- Max clock frequency (MHz)     : 1450
```

Ex1: Basic Query (gen9)

- Debug information
  - OpenCL backend (GPU default)

```
$ LIBOMPTARGET_DEBUG=1 LIBOMPTARGET_PLUGIN=OPENCL ./query.x

Libomptarget --> Init target library!
Libomptarget --> Callback to __tgt_register_ptask_services with handlers 0x00007fb1fe710e40
0x00007fb1fe711940
Libomptarget --> Initialized OMPT
Libomptarget --> Loading RTLs...
Libomptarget --> Checking user-specified plugin 'libomptarget.rtl.opencl.so'...
Libomptarget --> Loading library 'libomptarget.rtl.opencl.so'...
Target OPENCL RTL --> Init OpenCL plugin!
Target OPENCL RTL --> Target device type is set to GPU
Libomptarget --> Successfully loaded library 'libomptarget.rtl.opencl.so'!
```

  - OpenCL backend (CPU)

```
$ LIBOMPTARGET_DEBUG=1 LIBOMPTARGET_PLUGIN=OPENCL LIBOMPTARGET_DEVICETYPE=CPU ./query.x

Target OPENCL RTL --> Init OpenCL plugin!
Target OPENCL RTL --> ENV: LIBOMPTARGET_DEVICETYPE=CPU
Target OPENCL RTL --> Target device type is set to CPU
Libomptarget --> Successfully loaded library 'libomptarget.rtl.opencl.so'!
```

Ex1: Basic Query (gen9)

- Debug information (cont)
  - OpenCL backend (CPU)

```
$ LIBOMPTARGET_DEBUG=1 LIBOMPTARGET_PLUGIN=OPENCL LIBOMPTARGET_DEVICETYPE=CPU ./query.x

Target OPENCL RTL --> Device Properties:
Target OPENCL RTL --> -- Name                          : 11th Gen Intel(R) Core(TM) i9-11900KB @ 3.30GHz
Target OPENCL RTL --> -- PCI ID                        : 0x806d1
Target OPENCL RTL --> -- Number of total EUs           : 16
Target OPENCL RTL --> -- Number of threads per EU      : 2
Target OPENCL RTL --> -- Number of EUs per subslice    : 8
Target OPENCL RTL --> -- Number of subslices per slice: 1
Target OPENCL RTL --> -- Number of slices              : 1
Target OPENCL RTL --> -- Local memory size (bytes)     : 32768
Target OPENCL RTL --> -- Global memory size (bytes)    : 33182294016
Target OPENCL RTL --> -- Cache size (bytes)            : 262144
Target OPENCL RTL --> -- Max clock frequency (MHz)     : 3300
Target OPENCL RTL --> -- Max workgroup size            : 8192
Target OPENCL RTL --> -- Max allocation size (bytes)   : 16591147008
```

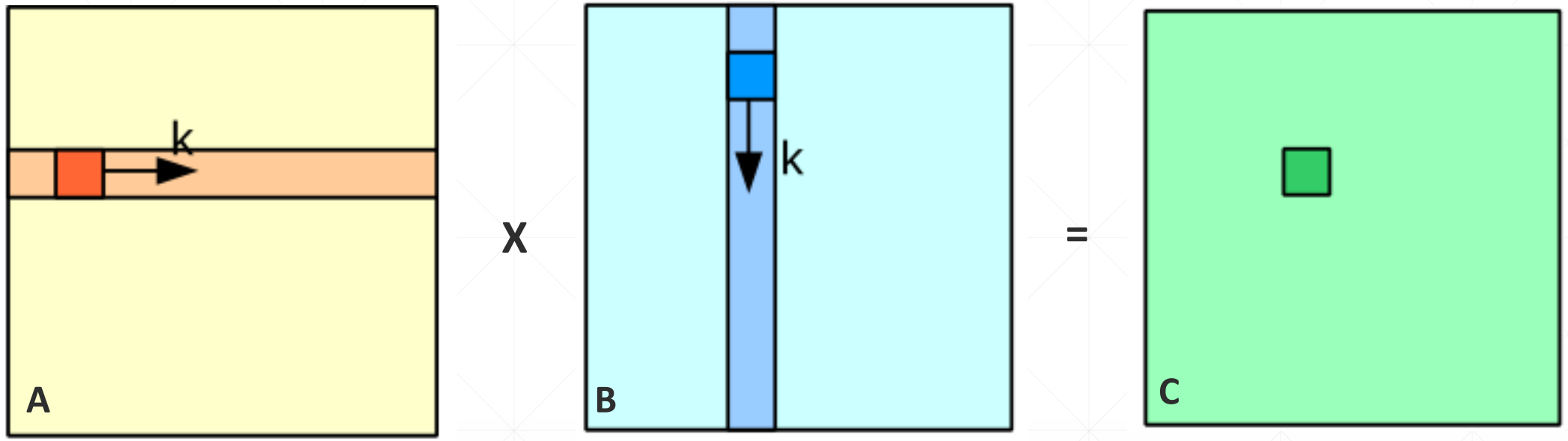
  - Number of subslice: CPU sockets
  - Number of EU: CPU cores
  - Number of threads per EU: hyper-threading

Ex2: Thread Creation

```
program hello
  use omp_lib
  implicit none

  integer :: num_devices,nteams,nthreads
  logical :: initial_device

  num_devices = omp_get_num_devices()
  print *, "Number of available devices", num_devices

  !$omp target map(nteams,nthreads)
  !$omp teams num_teams(2) thread_limit(3)
  !$omp parallel
    initial_device = omp_is_initial_device()
    nteams= omp_get_num_teams()
    nthreads= omp_get_num_threads()
  !$omp end parallel
  !$omp end teams
  !$omp end target
    if (initial_device) then
      write(*,*) "Running on host"
    else
      write(*,'(A,I4,A,I4,A)') "Running on device with ",nteams, " teams in total and ", nthreads, " threads in each team"
    end if

end program
```

```
Number of available devices           1
Running on device with    2 teams in total and    3 threads in each team
```

Ex3: Matrix Multiplication



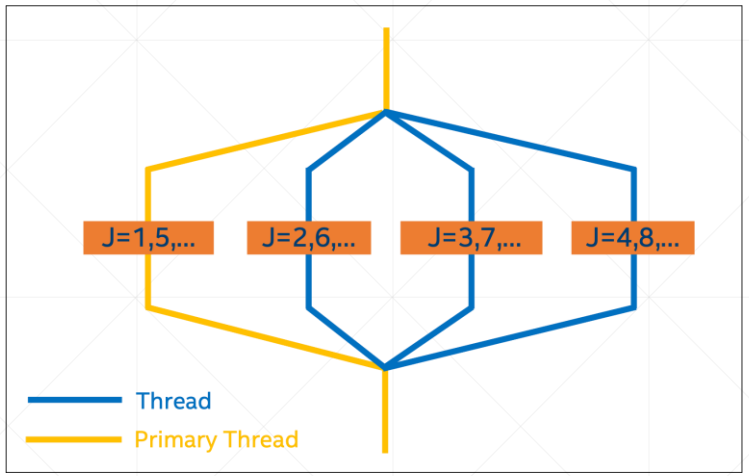

```
do j=1,n
   do i=1,n
      do k=1,n
         A(i,j) = C(i,j) + A(i,k) * B(k,j)
      enddo
  enddo
enddo
```

Ex3: Matrix Multiplication (Serial)

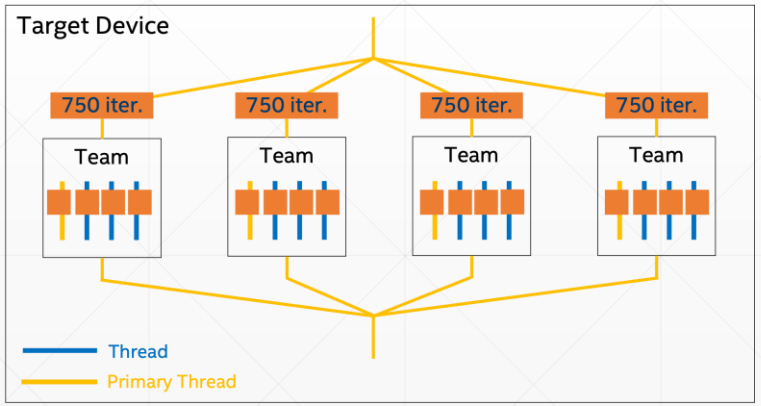
```
program matrix_multiply
   implicit none

   integer
:: i, j, k, myid, m, n
   real, allocatable, dimension(:,:) :: a, b, c,
c_serial

   n = 2600
   allocate( a(n,n), b(n,n), c(n,n),
c_serial(n,n))

   print *, 'matrix size ', n

   ! Initialize matrices
   do j=1,n
     do i=1,n
       a(i,j) = i + j - 1
       b(i,j) = i - j + 1
     enddo
   enddo

   c = 0.0

   c_serial = 0.0
```

```
   ! serial compute matrix multiplication
   do j=1,n
     do i=1,n
       do k=1,n
         c_serial(i,j) = c_serial(i,j) + a(i,k) * b(k,j)
       enddo
     enddo
   enddo

   ! parallel compute matrix multiplication.
   (To be implemented)

   ! verify result
   do j=1,n
     do i=1,n
       if (c_serial(i,j) .ne. c(i,j)) then
         print *,'FAILED'
         exit
       endif
     enddo
   enddo

   print *,'PASSED'

end program matrix_multiply
```

Ex3: Matrix Multiplication




```
! OpenMP parallelization
!$omp parallel do shared(a, b, c, n), private(i, j, k)
do j=1,n
   do i=1,n
       do k=1,n
           c(i,j) = c(i,j) + a(i,k) * b(k,j)
       enddo
    enddo
enddo
!$omp end parallel do
```




```
! OpenMP offload
!$omp target teams map(to: a, b) map(tofrom: c)
!$omp distribute parallel do SIMD private(j, i, k)
do j=1,n
   do i=1,n
      do k=1,n
          c(i,j) = c(i,j) + a(i,k) * b(k,j)
      enddo
   enddo
enddo
!$omp end target teams
```

Ex3: Matrix Multiplication

```
$ ifx -xhost mm_serial.f90 -o mm_serial.x
$ ./mm_serial.x
matrix size        4096
PASSED
```

```
$ ifx -xhost -qopenmp mm_omp.f90 -o mm_omp.x
$ time -p ./mm_omp.x
matrix size        4096
Number of procs is         16
PASSED
```

```
$ ifx -xhost -qopenmp -fopenmp-targets=spir64 mm_offload.f90 -o mm_omp.x
$ time -p ./mm_offload.x
matrix size        4096
Number of CPU procs is         1
Number of OpenMP Device Available:         1
Running on GPU
PASSED
```

Ex3: Matrix Multiplication

- Profiling

```
$ LIBOMPTARGET_PLUGIN_PROFILE=T ./mm_offload.x

================================================================================================================================
LIBOMPTARGET_PLUGIN_PROFILE(LEVEL0) for OMP DEVICE(0) Intel(R) UHD Graphics, Thread 0
--------------------------------------------------------------------------------------------------------------------------------
Kernel 0                  : __omp_offloading_35_3770474d_MAIN___l38
--------------------------------------------------------------------------------------------------------------------------------
                          : Host Time (msec)                          Device Time (msec)
Name                      :      Total   Average       Min       Max     Total   Average       Min       Max     Count
--------------------------------------------------------------------------------------------------------------------------------
Compiling                 :     166.47    166.47    166.47    166.47      0.00      0.00      0.00      0.00      1.00
DataAlloc                 :       5.16      0.37      0.00      1.73      0.00      0.00      0.00      0.00     14.00
DataRead (Device to Host) :       1.57      1.57      1.57      1.57      0.00      0.00      0.00      0.00      1.00
DataWrite (Host to Device):       4.59      0.51      0.00      1.57      0.00      0.00      0.00      0.00      9.00
Kernel 0                  :    3152.39   3152.39   3152.39   3152.39   3147.24   3147.24   3147.24   3147.24      1.00
Linking                   :       0.00      0.00      0.00      0.00      0.00      0.00      0.00      0.00      1.00
OffloadEntriesInit        :       0.97      0.97      0.97      0.97      0.00      0.00      0.00      0.00      1.00
================================================================================================================================
```

THANK YOU