www.locuz.com

Intel oneAPI Series - Deep Dive (Live Demo and Hands-on): oneAPI and DPC++ on Intel DevCloud

Presented By: Mandeep Kumar

Converge to the Cloud

Agenda

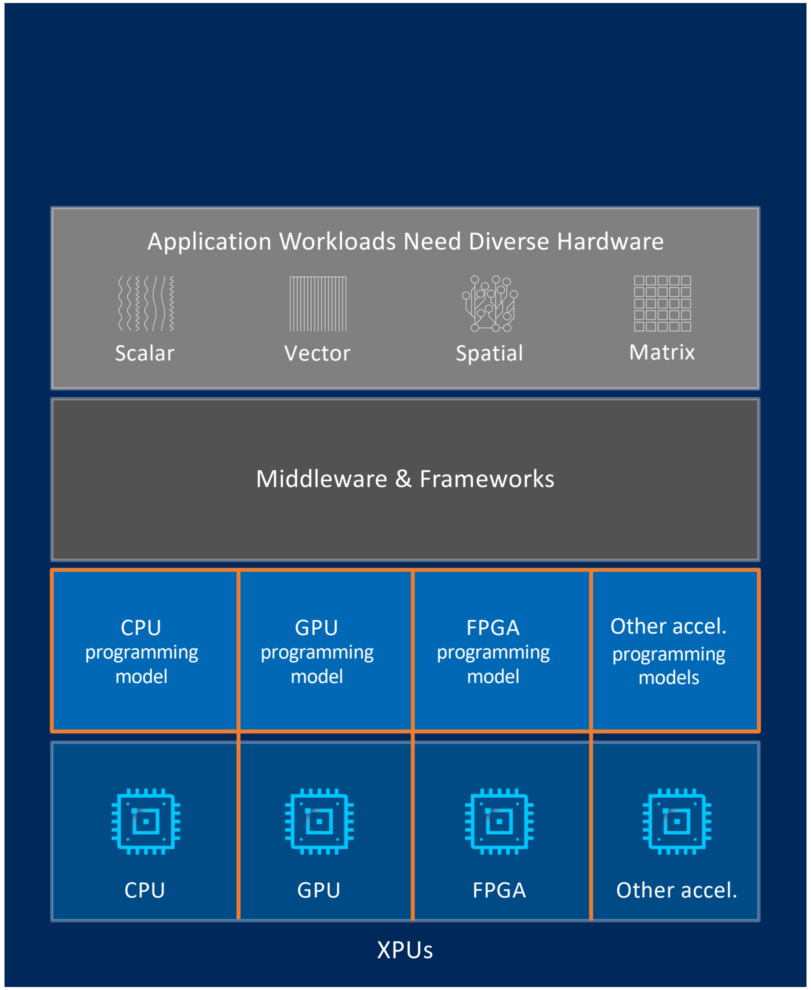
- Overview of Intel oneAPI and Data Parallel C++
- Introduction to the Intel DevCloud
- Setup of Intel DevCloud and JupyterLab Environment
- DPC++ Program Structure
- Demonstration of Intel VTune Profiler on Intel DevCloud
- Demonstration of Intel DPC++ Compatibility Tool on Intel DevCloud

Learning Objectives

- Articulate how oneAPI can help to solve the challenges of programming in a heterogeneous world
- Understand the DPC++ language and programming model
- Profile a DPC++ application using Intel VTune Profiler on Intel DevCloud
- Learn how to migrate CUDA code to Data Parallel C++ using the Intel DPC++ Compatibility tool

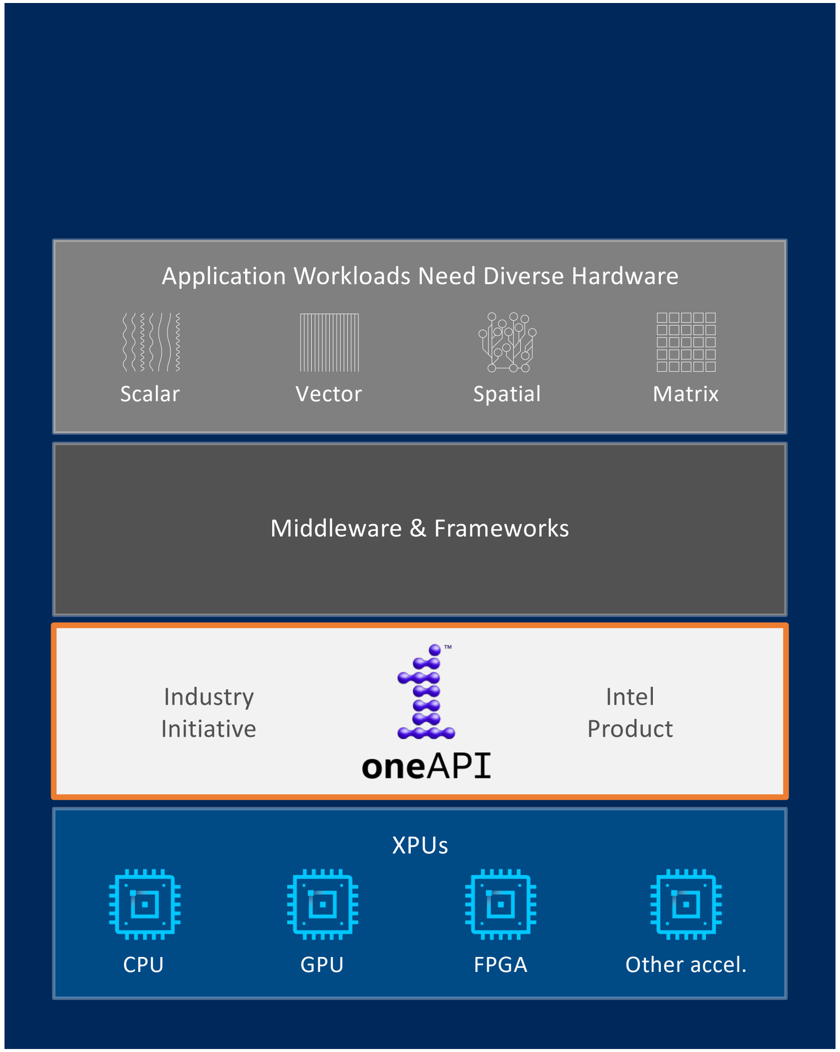
Programming Challenges for Multiple Architectures

- Growth in specialized workloads
- Variety of data-centric hardware required
- Separate programming models and toolchains for each architecture are required today
- Software development complexity limits freedom of architectural choice



Introducing oneAPI

- Cross-architecture programming that delivers freedom to choose the best hardware
- Based on industry standards and open specifications
- Exposes cutting-edge performance features of latest hardware
- Compatible with existing high-performance languages and programming models including C++, OpenMP, Fortran, and MPI



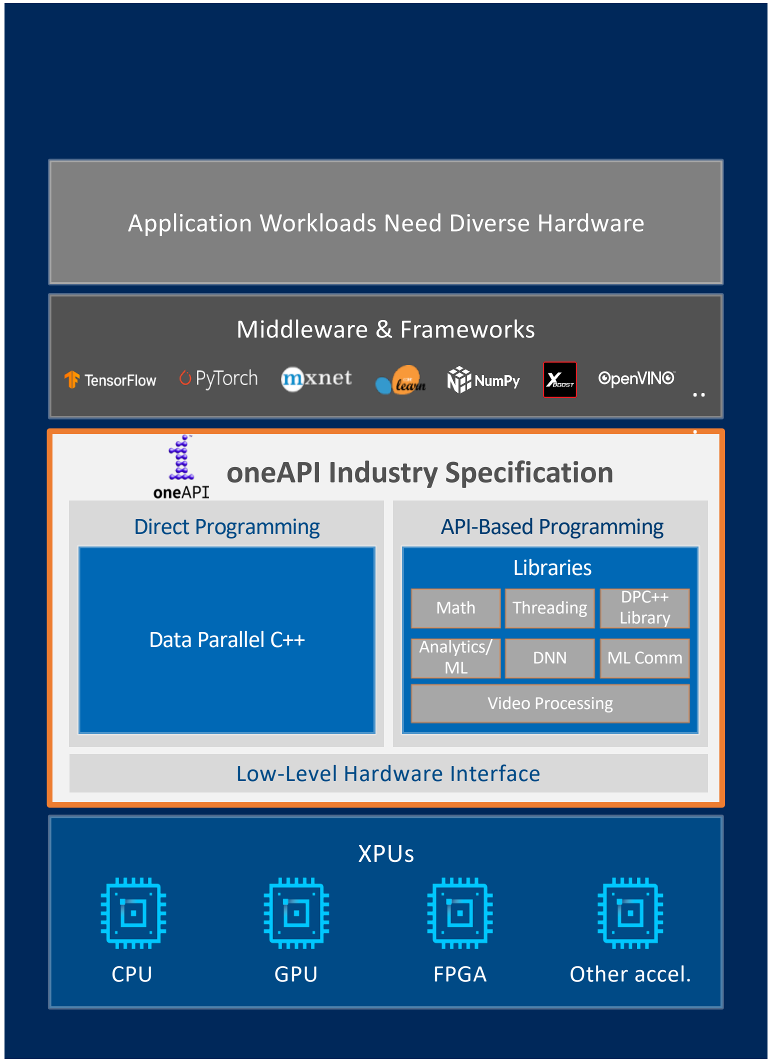
oneAPI Industry Initiative: Break the Chains of Proprietary Lock-in

- A cross-architecture language based on C++ and SYCL standards
- Powerful libraries designed for acceleration of domain-specific functions
- Low-level hardware abstraction layer
- **Open to promote community and industry collaboration**
- **Enables code reuse across architectures and vendors**

oneAPI

The productive, smart path to freedom for accelerated computing from the economic and technical burdens of proprietary programming models

visit oneapi.com for more details



Application Workloads Need Diverse Hardware

Middleware & Frameworks

TensorFlow PyTorch mxnet learn NumPy XBOOST OpenVINO ..

oneAPI Industry Specification

Direct Programming

Data Parallel C++

API-Based Programming

Libraries

Math | Threading | DPC++ Library

Analytics/ ML | DNN | ML Comm

Video Processing

Low-Level Hardware Interface

XPUs

CPU | GPU | FPGA | Other accel.

Data Parallel C++: Standards-based, Cross-architecture Language

Parallelism, productivity and performance for CPUs and Accelerators

- Delivers accelerated computing by exposing hardware features
- Allows code reuse across hardware targets, while permitting custom tuning for specific accelerators
- Provides an open, cross-industry solution to single architecture proprietary lock-in

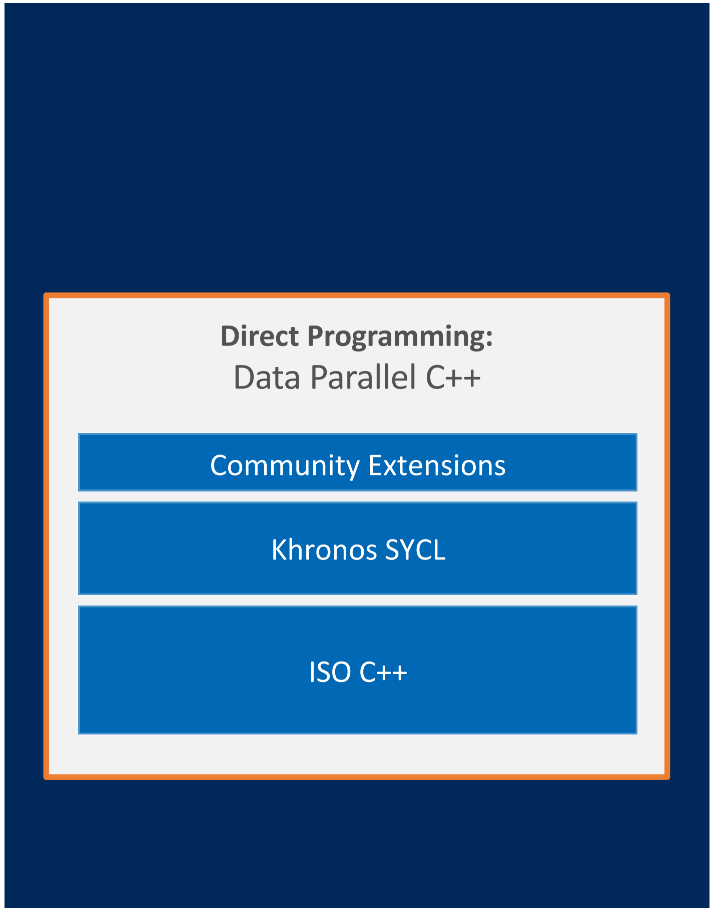
Based on C++ and SYCL

- Delivers C++ productivity benefits, using common, familiar C and C++ constructs
- Incorporates SYCL from the Khronos Group to support data parallelism and heterogeneous programming

Community Project to drive language enhancements

- Provides extensions to simplify data parallel programming
- Continues evolution through open and cooperative development

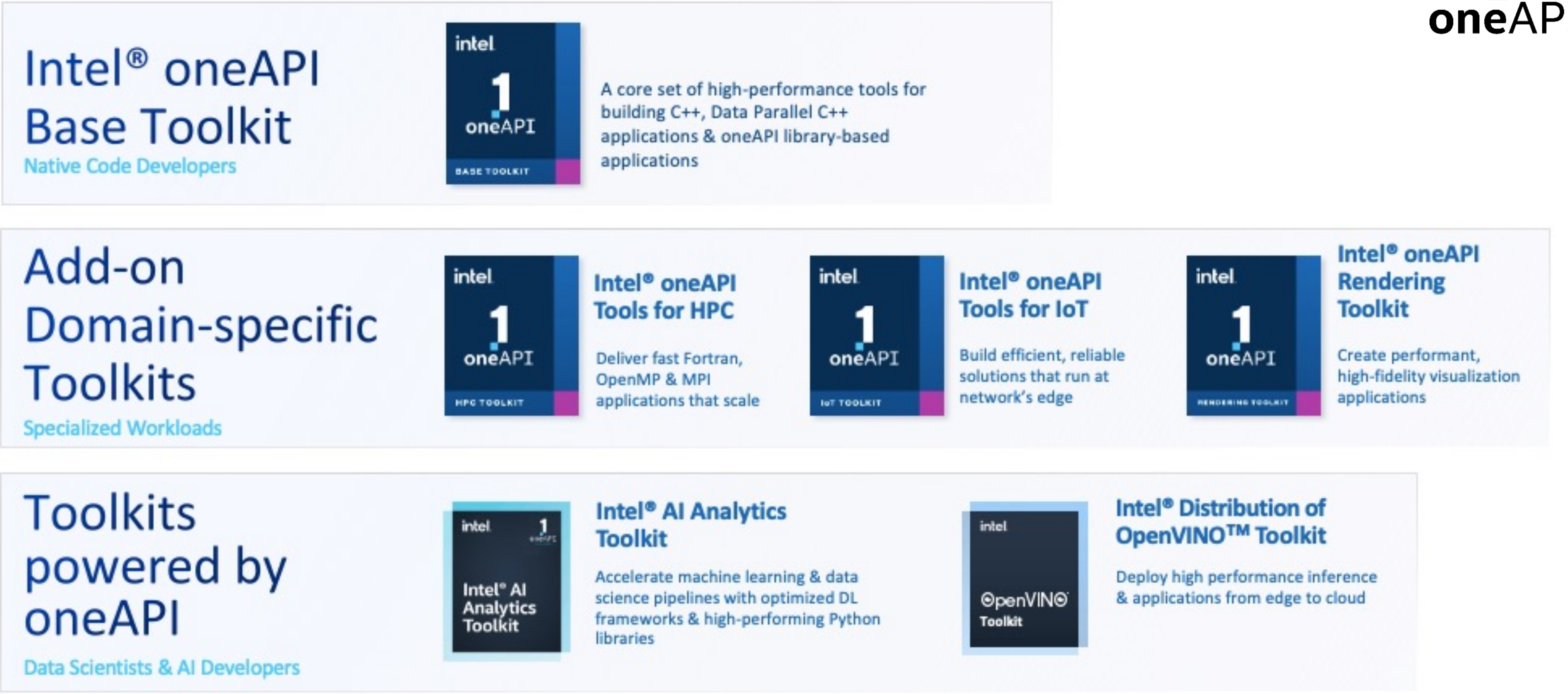
Apply your skills to the next innovation, not rewriting software for the next hardware platform



The open source and Intel DPC++/C++ compiler supports Intel CPUs, GPUs, and FPGAs.
Codeplay announced a DPC++ complier that targets Nvidia GPUs.

Intel oneAPI Toolkits

A complete set of proven developer tools expanded from CPU to XPU



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

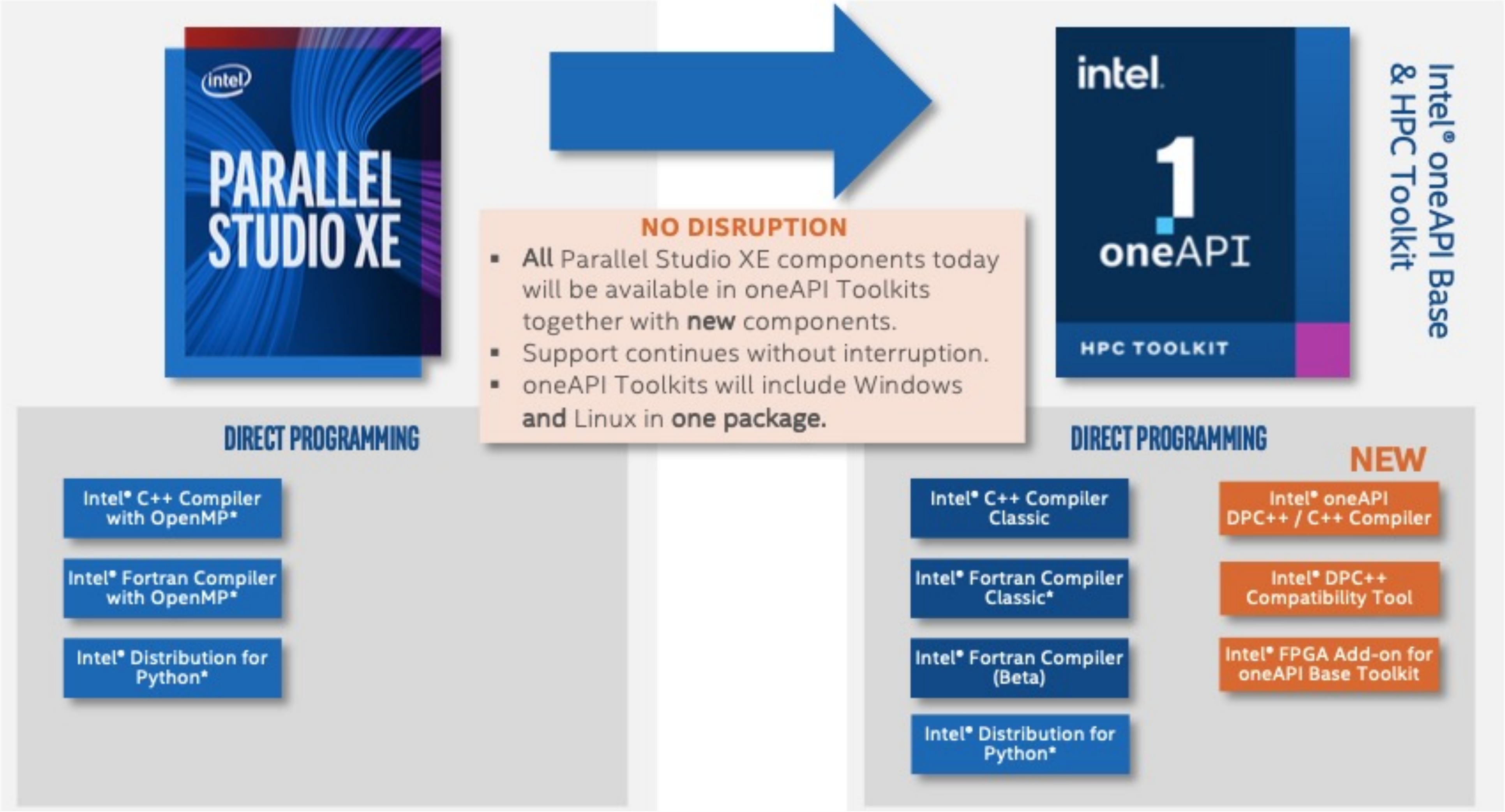
Intel® AI Analytics Toolkit

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

Transition from Parallel Studio XE with no Disruption



NO DISRUPTION

- **All** Parallel Studio XE components today will be available in oneAPI Toolkits together with **new** components.
- Support continues without interruption.
- oneAPI Toolkits will include Windows **and** Linux in **one package.**

DIRECT PROGRAMMING

- Intel® C++ Compiler with OpenMP*
- Intel® Fortran Compiler with OpenMP*
- Intel® Distribution for Python*

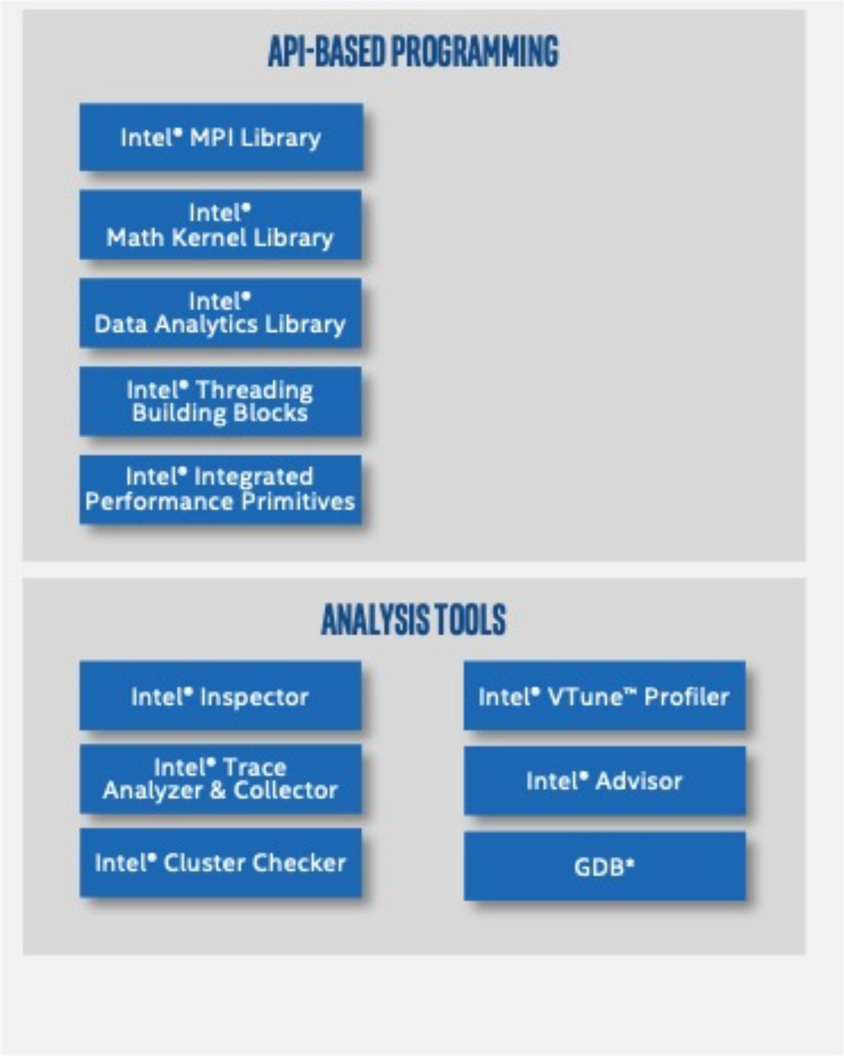
DIRECT PROGRAMMING

- Intel® C++ Compiler Classic
- Intel® Fortran Compiler Classic*
- Intel® Fortran Compiler (Beta)
- Intel® Distribution for Python*

NEW

- Intel® oneAPI DPC++ / C++ Compiler
- Intel® DPC++ Compatibility Tool
- Intel® FPGA Add-on for oneAPI Base Toolkit

Transition from Parallel Studio XE with no Disruption

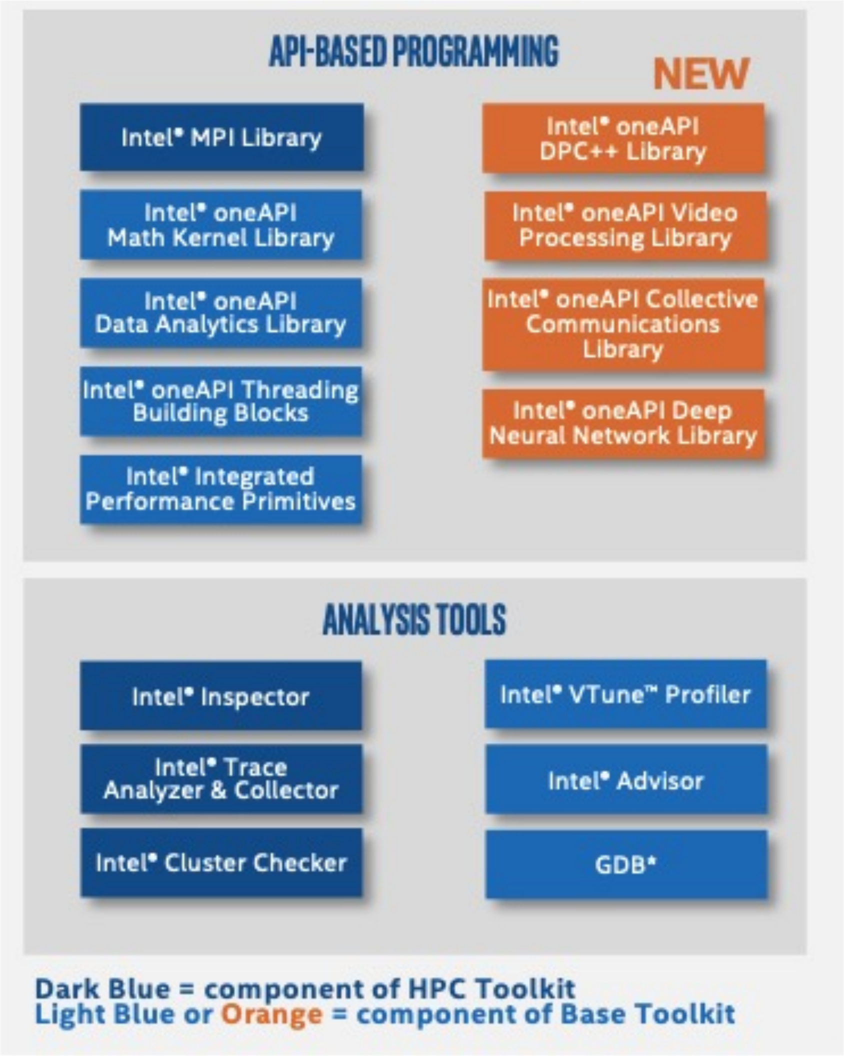


API-BASED PROGRAMMING

- Intel® MPI Library
- Intel® Math Kernel Library
- Intel® Data Analytics Library
- Intel® Threading Building Blocks
- Intel® Integrated Performance Primitives

ANALYSIS TOOLS

- Intel® Inspector
- Intel® Trace Analyzer & Collector
- Intel® Cluster Checker
- Intel® VTune™ Profiler
- Intel® Advisor
- GDB*



API-BASED PROGRAMMING

- Intel® MPI Library
- Intel® oneAPI Math Kernel Library
- Intel® oneAPI Data Analytics Library
- Intel® oneAPI Threading Building Blocks
- Intel® Integrated Performance Primitives

NEW

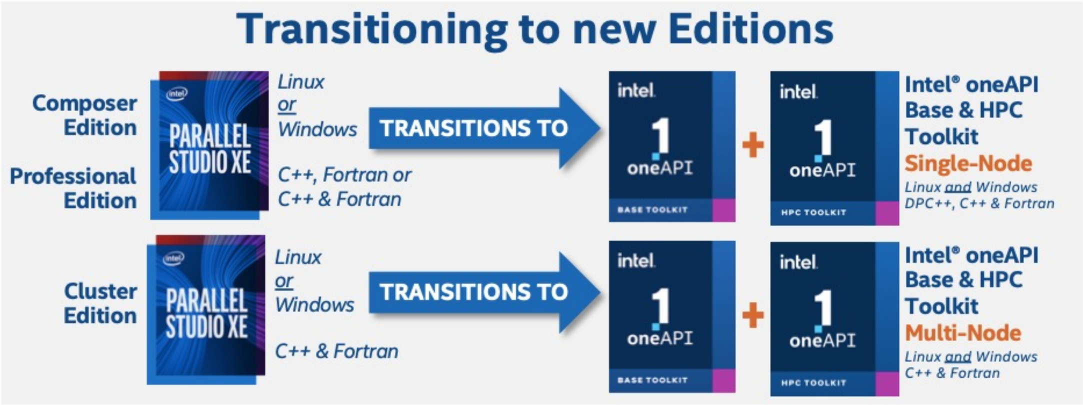
- Intel® oneAPI DPC++ Library
- Intel® oneAPI Video Processing Library
- Intel® oneAPI Collective Communications Library
- Intel® oneAPI Deep Neural Network Library

ANALYSIS TOOLS

- Intel® Inspector
- Intel® Trace Analyzer & Collector
- Intel® Cluster Checker
- Intel® VTune™ Profiler
- Intel® Advisor
- GDB*

Dark Blue = component of HPC Toolkit
Light Blue or Orange = component of Base Toolkit

Transition from Parallel Studio XE with no Disruption



Transitioning to new Editions

Composer Edition

Professional Edition

Linux or Windows

C++, Fortran or C++ & Fortran

TRANSITIONS TO

Intel® oneAPI Base & HPC Toolkit Single-Node

Linux and Windows DPC++, C++ & Fortran

Cluster Edition

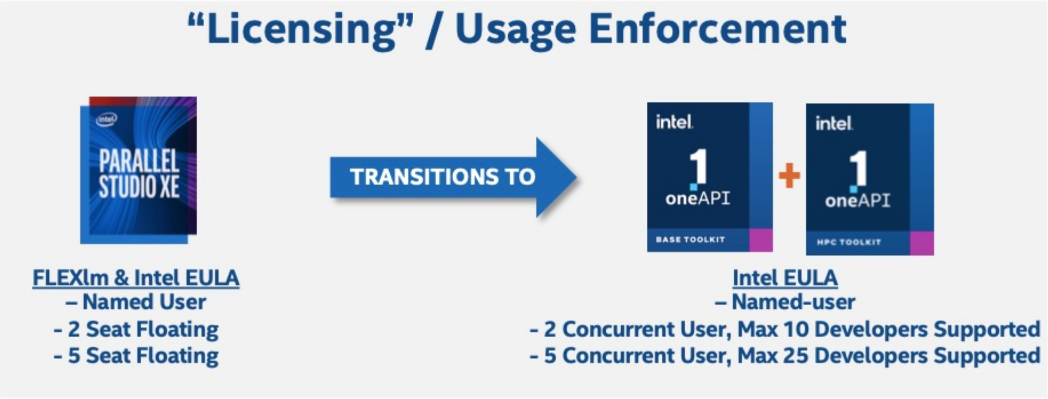
Linux or Windows

C++ & Fortran

TRANSITIONS TO

Intel® oneAPI Base & HPC Toolkit Multi-Node

Linux and Windows C++ & Fortran

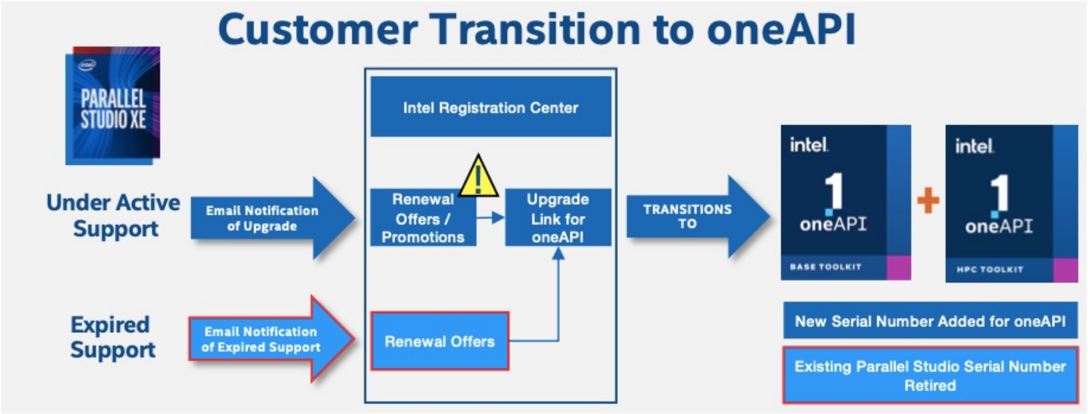


"Licensing" / Usage Enforcement

TRANSITIONS TO

FLEXlm & Intel EULA
– Named User
- 2 Seat Floating
- 5 Seat Floating

Intel EULA
– Named-user
- 2 Concurrent User, Max 10 Developers Supported
- 5 Concurrent User, Max 25 Developers Supported



Customer Transition to oneAPI

Under Active Support

Email Notification of Upgrade

Expired Support

Email Notification of Expired Support

Intel Registration Center

Renewal Offers / Promotions

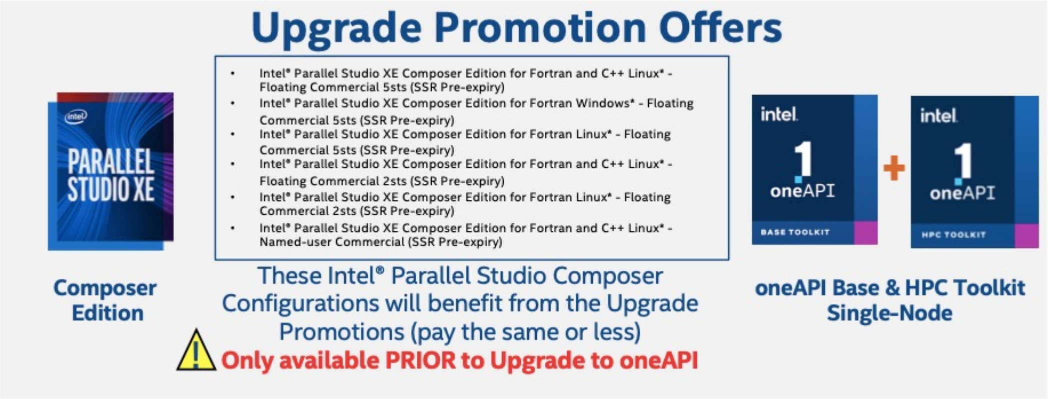
Upgrade Link for oneAPI

Renewal Offers

TRANSITIONS TO

New Serial Number Added for oneAPI

Existing Parallel Studio Serial Number Retired



Upgrade Promotion Offers

- Intel® Parallel Studio XE Composer Edition for Fortran and C++ Linux* - Floating Commercial 5sts (SSR Pre-expiry)
- Intel® Parallel Studio XE Composer Edition for Fortran Windows* - Floating Commercial 5sts (SSR Pre-expiry)
- Intel® Parallel Studio XE Composer Edition for Fortran Linux* - Floating Commercial 5sts (SSR Pre-expiry)
- Intel® Parallel Studio XE Composer Edition for Fortran and C++ Linux* - Floating Commercial 2sts (SSR Pre-expiry)
- Intel® Parallel Studio XE Composer Edition for Fortran Linux* - Floating Commercial 2sts (SSR Pre-expiry)
- Intel® Parallel Studio XE Composer Edition for Fortran and C++ Linux* - Named-user Commercial (SSR Pre-expiry)

Composer Edition

These Intel® Parallel Studio Composer Configurations will benefit from the Upgrade Promotions (pay the same or less)

⚠ **Only available PRIOR to Upgrade to oneAPI**

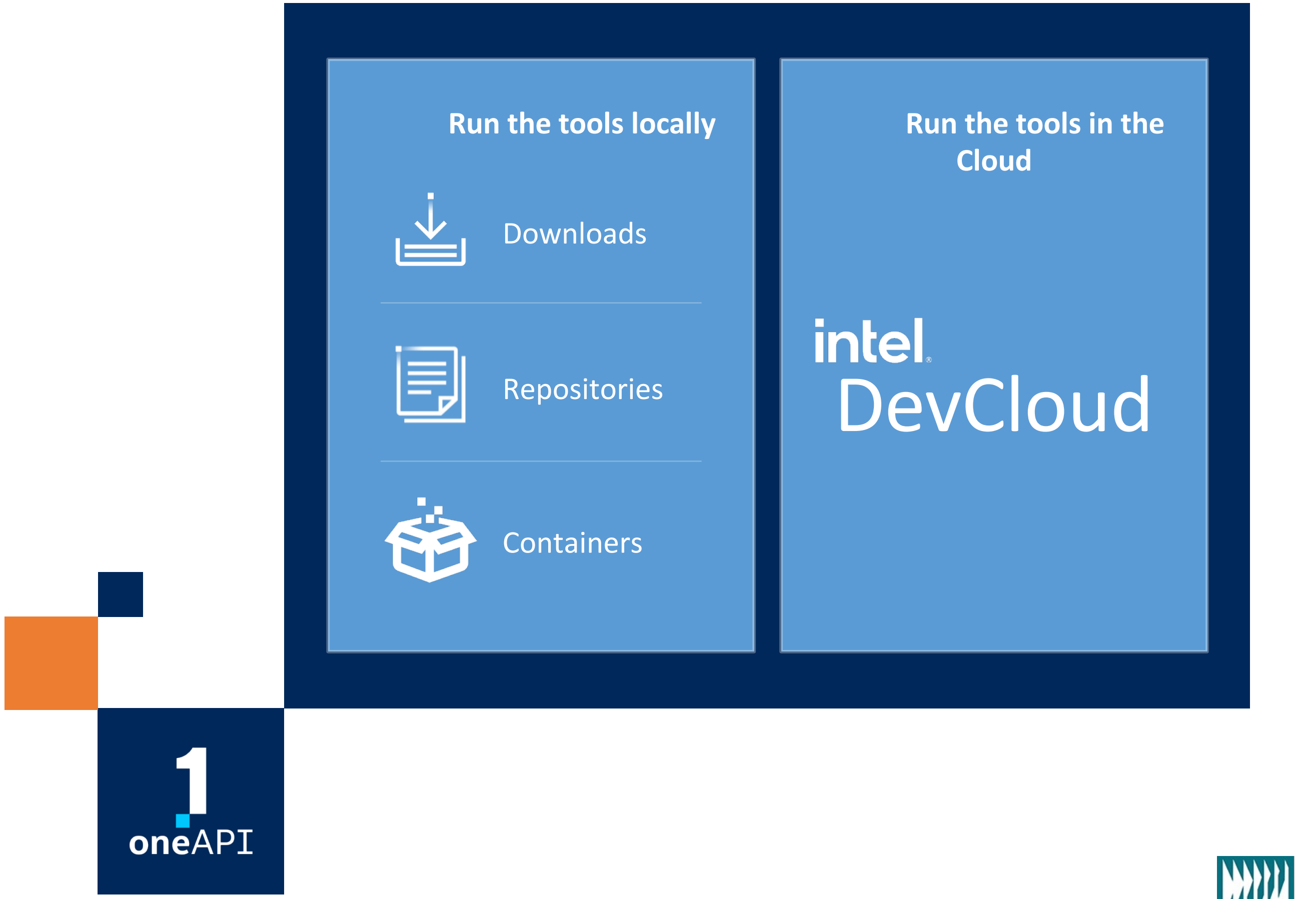
oneAPI Base & HPC Toolkit Single-Node

Intel oneAPI Toolkits Free Availability

Get Started Quickly

Code Samples, Quick-start Guides, Webinars, Training

software.intel.com/oneapi



Run the tools locally

Downloads

Repositories

Containers

Run the tools in the Cloud

intel DevCloud

oneAPI Available on: Intel DevCloud

A development sandbox to develop, test and run workloads across a range of Intel CPUs, GPUs, and FPGAs using Intel’s oneAPI software.

Get Up & Running In Seconds!

software.intel.com/devcloud/oneapi

intel DevCloud

1 Minute to Code

No Hardware Acquisition

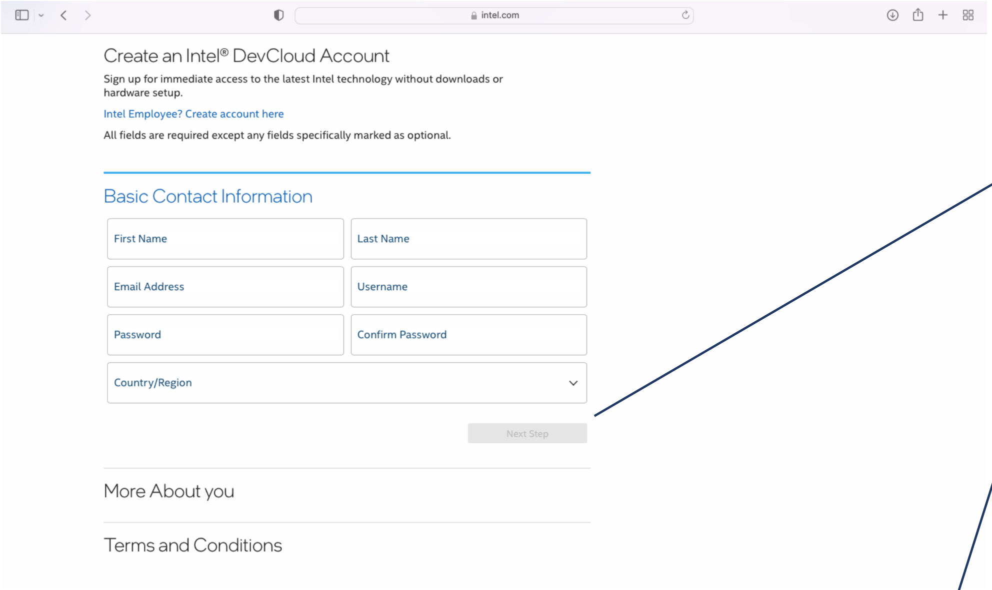
No Download, Install or Configuration

Easy Access to Samples & Tutorials

Support for Jupyter Notebooks, Visual Studio Code

Create an Intel DevCloud Account

software.intel.com/devcloud/oneapi



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

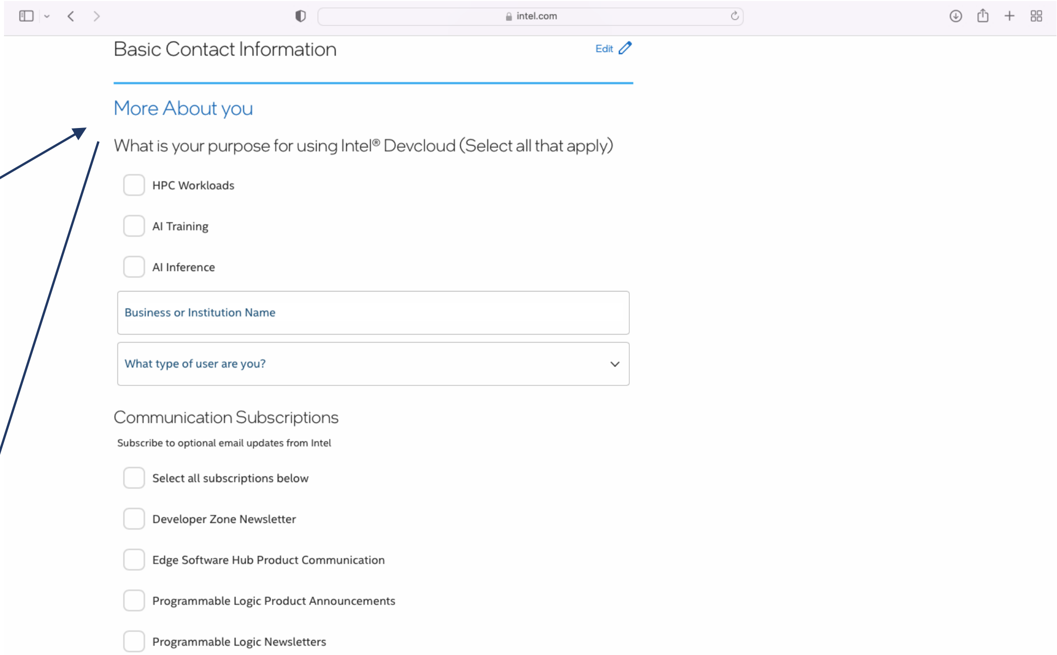
All fields are required except any fields specifically marked as optional.

Basic Contact Information

First Name	Last Name
Email Address	Username
Password	Confirm Password
Country/Region	

More About you

Terms and Conditions



Basic Contact Information

More About you

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

- □ HPC Workloads
- □ AI Training
- □ AI Inference

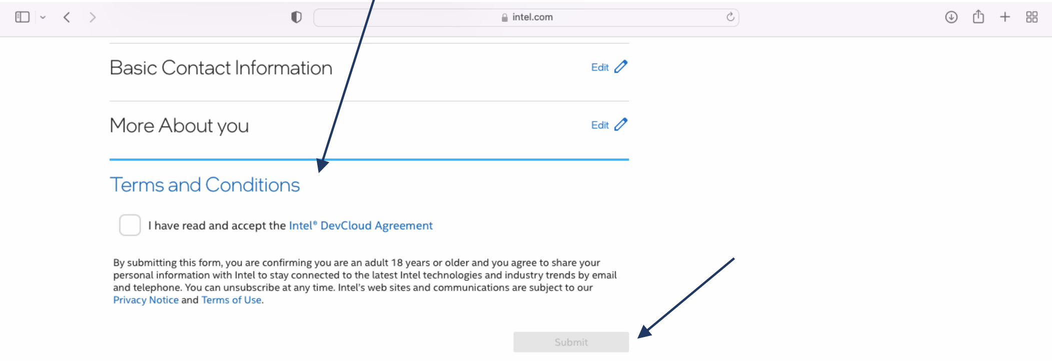
Business or Institution Name

What type of user are you?

Communication Subscriptions

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



Basic Contact Information

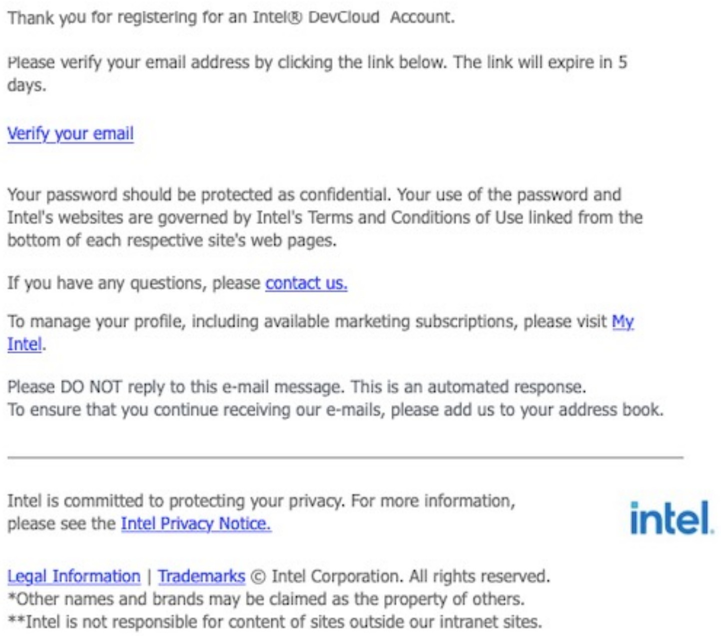
More About you

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.

Subject: Intel® DevCloud Account Confirmation - Email Verification



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.

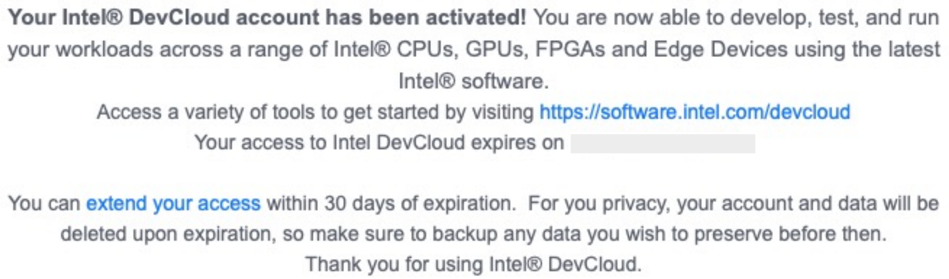
To manage your profile, including available marketing subscriptions, please visit My Intel.

Please DO NOT reply to this e-mail message. This is an automated response.
To ensure that you continue receiving our e-mails, please add us to your address book.

Intel is committed to protecting your privacy. For more information, please see the Intel Privacy Notice.

Legal Information | Trademarks © Intel Corporation. All rights reserved.
*Other names and brands may be claimed as the property of others.
**Intel is not responsible for content of sites outside our intranet sites.

Subject: Welcome to Intel® DevCloud



Your Intel® DevCloud account has been activated! You are now able to develop, test, and run your workloads across a range of Intel® CPUs, GPUs, FPGAs and Edge Devices using the latest Intel® software.

Access a variety of tools to get started by visiting https://software.intel.com/devcloud

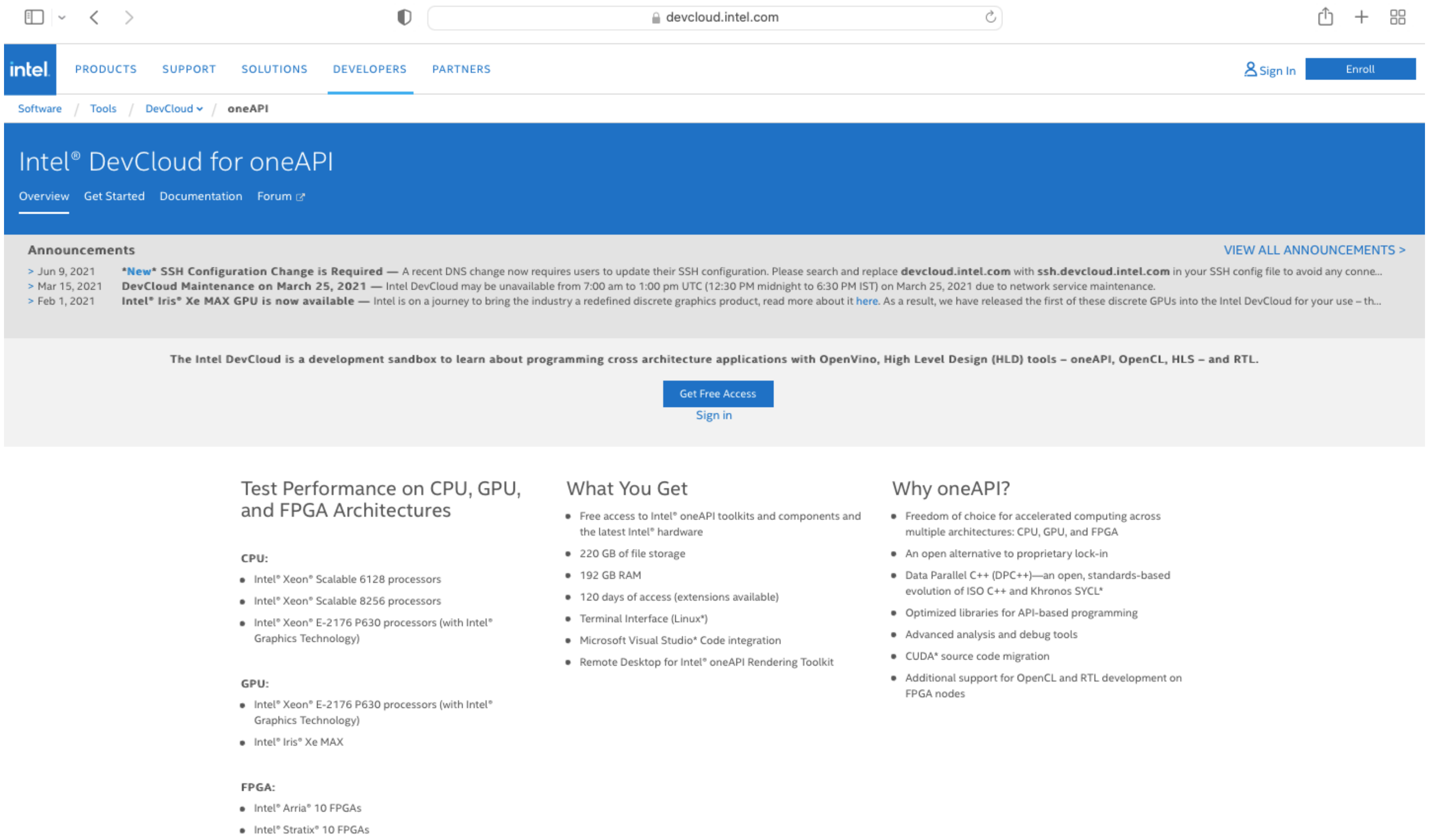
Your access to Intel DevCloud expires on

You can extend your access within 30 days of expiration. For you privacy, your account and data will be deleted upon expiration, so make sure to backup any data you wish to preserve before then.

Thank you for using Intel® DevCloud.

Access a variety of tools to get started

devcloud.intel.com/oneapi



Intel® DevCloud for oneAPI

Announcements

VIEW ALL ANNOUNCEMENTS >

> Jun 9, 2021 ***New* 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 conne...

> Mar 15, 2021 **DevCloud Maintenance on March 25, 2021** — Intel DevCloud may be unavailable from 7:00 am to 1:00 pm UTC (12:30 PM midnight to 6:30 PM IST) on March 25, 2021 due to network service maintenance.

> Feb 1, 2021 **Intel® Iris® Xe MAX GPU is now available** — Intel is on a journey to bring the industry a redefined discrete graphics product, read more about it here. As a result, we have released the first of these discrete GPUs into the Intel DevCloud for your use – th...

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.

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

FPGA:

- Intel® Arria® 10 FPGAs
- Intel® Stratix® 10 FPGAs

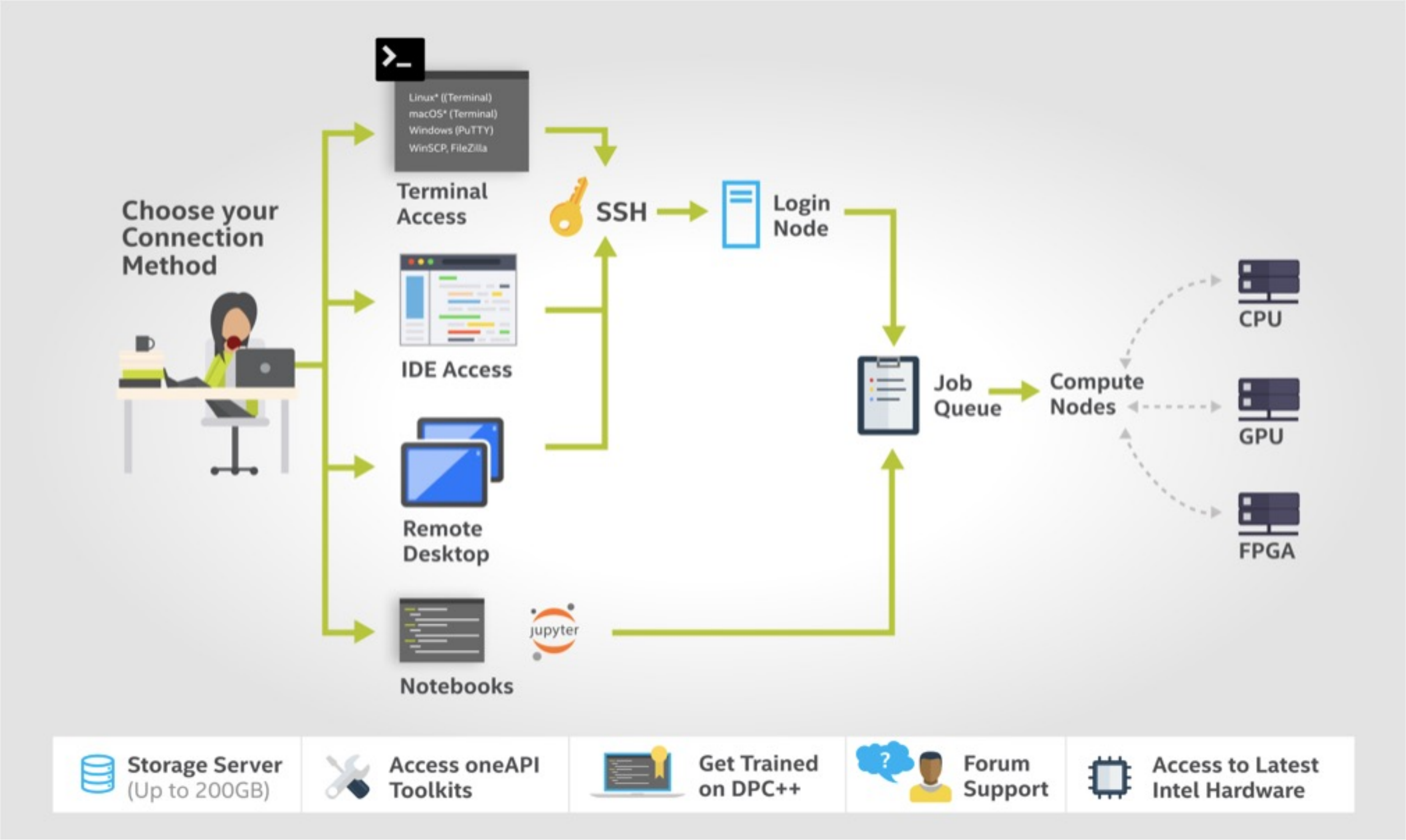
What You Get

- 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

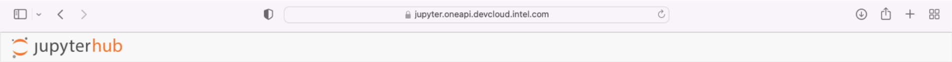
Choose your Connection Method

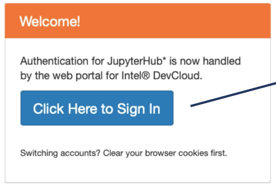


Setup of Intel DevCloud and JupyterLab Environment

Launch Jupyterlab and select Terminal

jupyter.oneapi.devcloud.intel.com



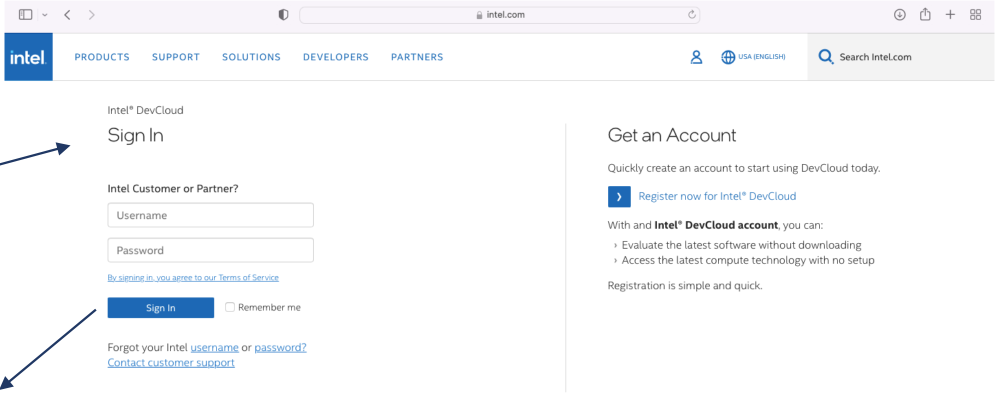


Welcome!

Authentication for JupyterHub* is now handled by the web portal for Intel® DevCloud.

Click Here to Sign In

Switching accounts? Clear your browser cookies first.



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

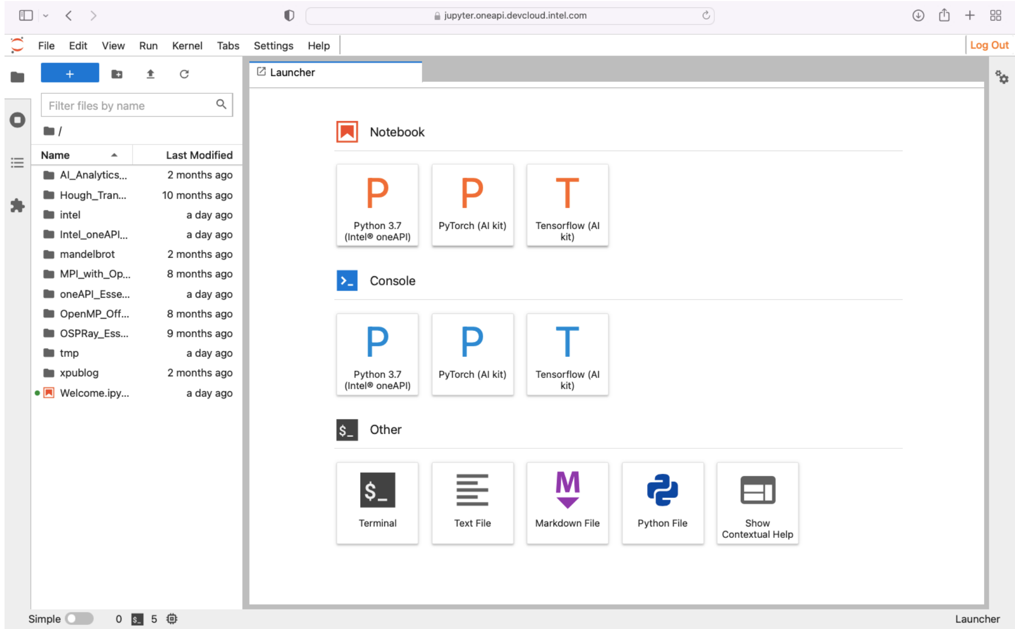
Quickly create an account to start using DevCloud today.

Register now for Intel® DevCloud

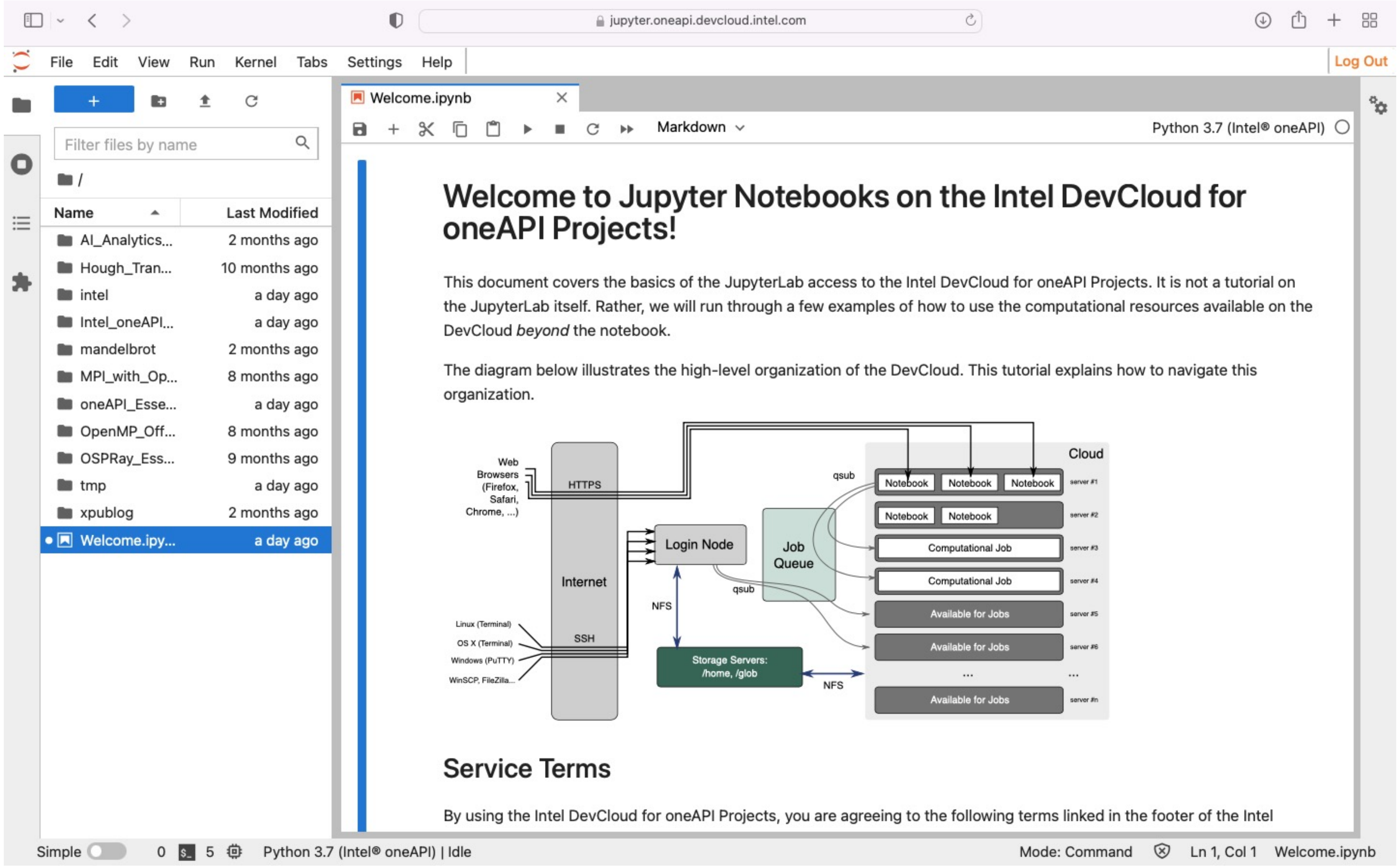
With and **Intel® DevCloud account**, you can:

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

Registration is simple and quick.



Name	Last Modified
AI_Analytics...	2 months ago
Hough_Tran...	10 months ago
intel	a day ago
Intel_oneAPI...	a day ago
mandelbrot	2 months ago
MPI_with_Op...	8 months ago
oneAPI_Esse...	a day ago
OpenMP_Off...	8 months ago
OSPRay_Ess...	9 months ago
tmp	a day ago
xpublog	2 months ago
Welcome.ipy...	a day ago



Name	Last Modified
AI_Analytics...	2 months ago
Hough_Tran...	10 months ago
intel	a day ago
Intel_oneAPI...	a day ago
mandelbrot	2 months ago
MPI_with_Op...	8 months ago
oneAPI_Esse...	a day ago
OpenMP_Off...	8 months ago
OSPRay_Ess...	9 months ago
tmp	a day ago
xpublog	2 months ago
Welcome.ipy...	a day ago

Welcome to Jupyter Notebooks on the Intel DevCloud for oneAPI Projects!

This document covers the basics of the JupyterLab access to the Intel DevCloud for oneAPI Projects. It is not a tutorial on the JupyterLab itself. Rather, we will run through a few examples of how to use the computational resources available on the DevCloud *beyond* the notebook.

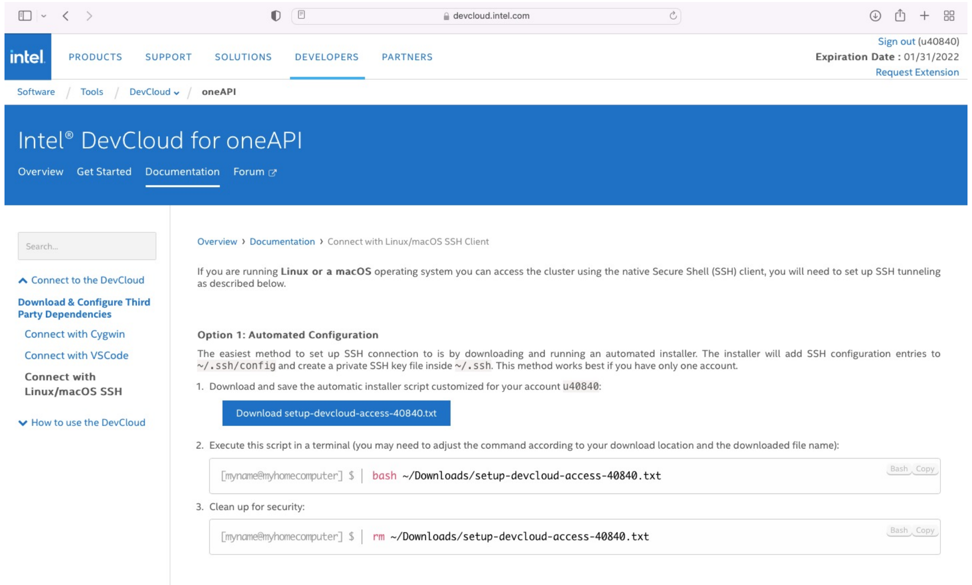
The diagram below illustrates the high-level organization of the DevCloud. This tutorial explains how to navigate this organization.

Service Terms

By using the Intel DevCloud for oneAPI Projects, you are agreeing to the following terms linked in the footer of the Intel

Connect with Linux/macOS SSH Client

devcloud.intel.com/oneapi/documentation/connect-with-ssh-linux-macos/



Intel® DevCloud for oneAPI

If you are running **Linux or a macOS** operating system you can access the cluster using the native Secure Shell (SSH) client, you will need to set up SSH tunneling as described below.

Option 1: Automated Configuration

The easiest method to set up SSH connection to is by downloading and running an automated installer. The installer will add SSH configuration entries to `~/.ssh/config` and create a private SSH key file inside `~/.ssh`. This method works best if you have only one account.

1. Download and save the automatic installer script customized for your account `u40840`:

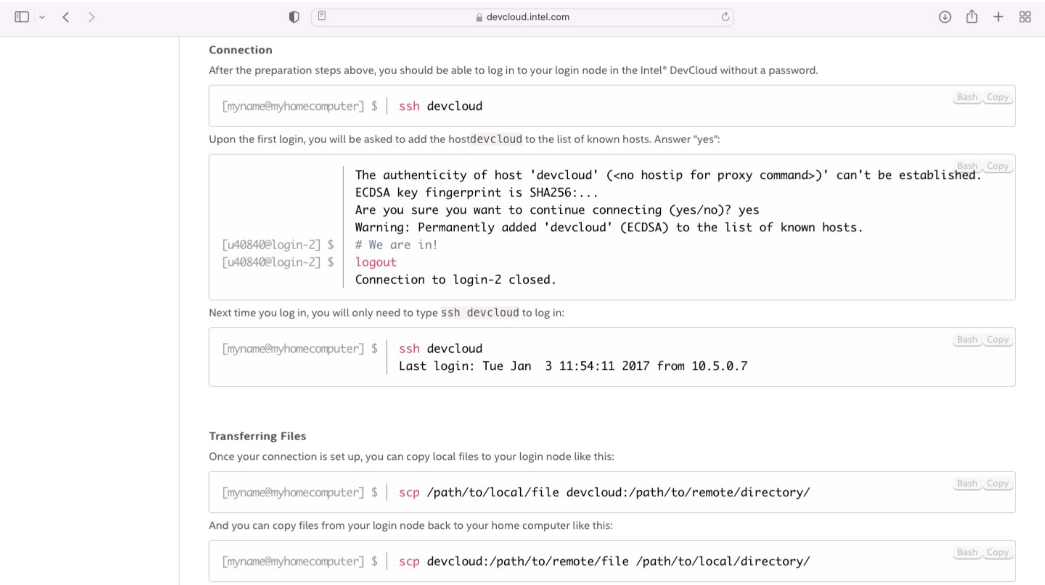
   Download setup-devcloud-access-40840.txt

2. Execute this script in a terminal (you may need to adjust the command according to your download location and the downloaded file name):

```
[myname@myhomecomputer] $ | bash ~/Downloads/setup-devcloud-access-40840.txt
```

3. Clean up for security:

```
[myname@myhomecomputer] $ | rm ~/Downloads/setup-devcloud-access-40840.txt
```



Connection

After the preparation steps above, you should be able to log in to your login node in the Intel® DevCloud without a password.

```
[myname@myhomecomputer] $ | ssh devcloud
```

Upon the first login, you will be asked to add the host`devcloud` to the list of known hosts. Answer "yes":

```
                          | The authenticity of host 'devcloud' (<no hostip for proxy command>)' can't be established.
                          | ECDSA key fingerprint is SHA256:...
                          | Are you sure you want to continue connecting (yes/no)? yes
                          | Warning: Permanently added 'devcloud' (ECDSA) to the list of known hosts.
[u40840@login-2] $        | # We are in!
[u40840@login-2] $        | logout
                          | Connection to login-2 closed.
```

Next time you log in, you will only need to type `ssh devcloud` to log in:

```
[myname@myhomecomputer] $ | ssh devcloud
                          | Last login: Tue Jan  3 11:54:11 2017 from 10.5.0.7
```

Transferring Files

Once your connection is set up, you can copy local files to your login node like this:

```
[myname@myhomecomputer] $ | scp /path/to/local/file devcloud:/path/to/remote/directory/
```

And you can copy files from your login node back to your home computer like this:

```
[myname@myhomecomputer] $ | scp devcloud:/path/to/remote/file /path/to/local/directory/
```

A Complete DPC++ Program

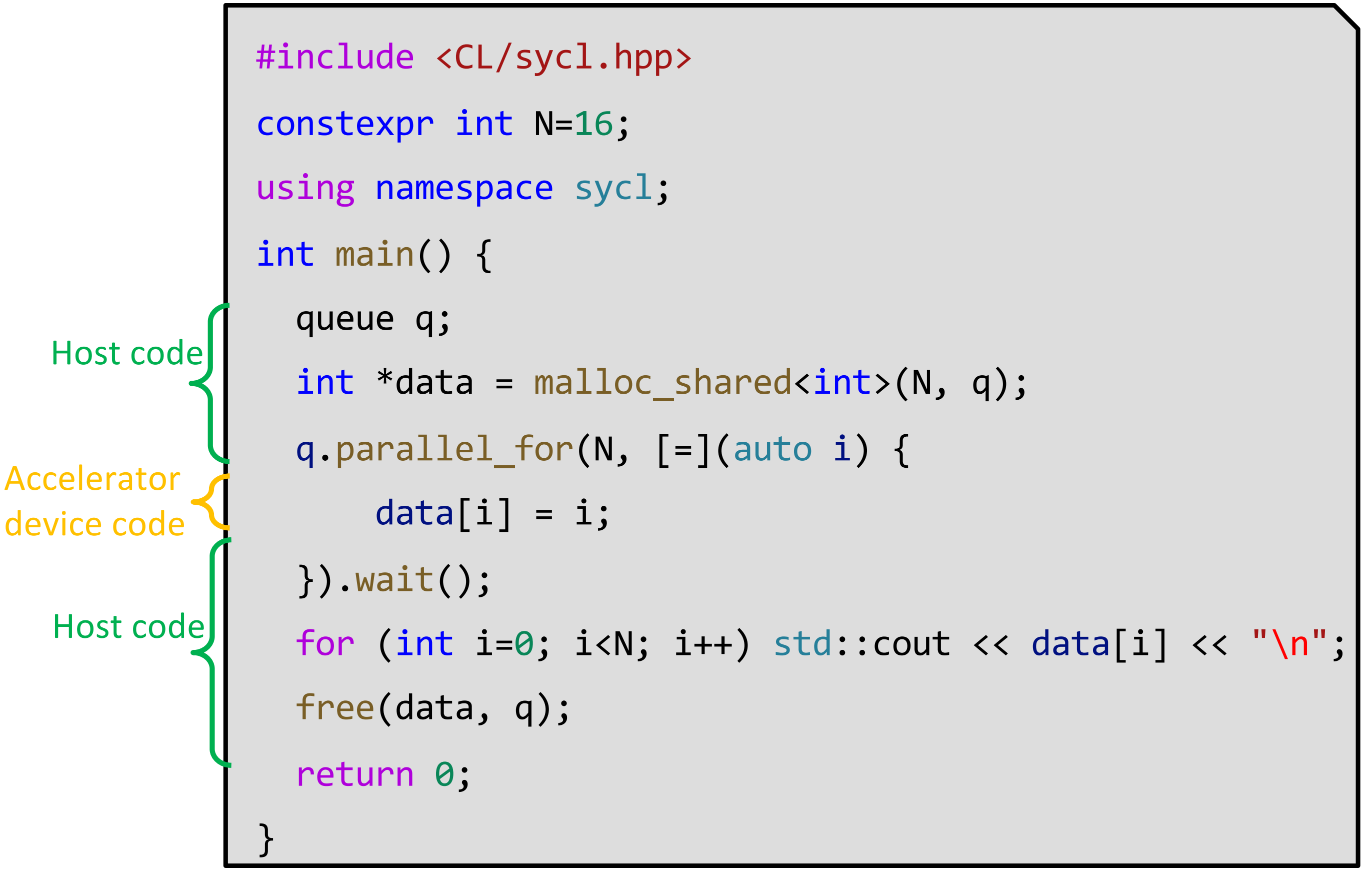
Single source

- Host code and heterogeneous accelerator kernels can be mixed in same source files

Familiar C++

- Library constructs add functionality, such as:

Construct	Purpose
queue	Work targeting
malloc_shared	Data management
parallel_for	Parallelism



```
#include <CL/sycl.hpp>
constexpr int N=16;
using namespace sycl;
int main() {
  queue q;
  int *data = malloc_shared<int>(N, q);
  q.parallel_for(N, [=](auto i) {
      data[i] = i;
  }).wait();
  for (int i=0; i<N; i++) std::cout << data[i] << "\n";
  free(data, q);
  return 0;
}
```

SYCL Classes

Device

- The **device** class represents the capabilities of the accelerators in a oneAPI system.
- The device class contains member functions for querying information about the device, which is useful for DPC++ programs where multiple devices are created.
- The function get_info gives information about the device:

```
queue q;
device my_device = q.get_device();
std::cout << "Device: " << my_device.get_info<info::device::name>() << std::endl;
```

Device Selector

- The **device_selector** class enables the runtime selection of a particular device to execute kernels based upon user-provided heuristics.
- The following code sample shows use of the standard device selectors (default_selector, cpu_selector, gpu_selector…) and a derived **device_selector**

```
default_selector selector;
// host_selector selector;
// cpu_selector selector;
// gpu_selector selector;
queue q(selector);
std::cout << "Device: " << q.get_device().get_info<info::device::name>() << std::endl;
```

Queue

- A queue **submits command groups** to be executed by the SYCL runtime
- Queue is a mechanism where work is submitted to a device.
- A Queue map to one device and multiple queues can be mapped to the same device.

```
queue q;


q.submit([&](handler& h) {
    // COMMAND GROUP CODE
});
```

Choosing Where Device Kernels Run

Work is submitted to queues

- Each queue is associated with exactly one device (e.g. a specific GPU or FPGA)
- You can:
  - Decide which device a queue is associated with (if you want)
  - Have as many queues as desired for dispatching work in heterogeneous systems

Create queue targeting any device:	queue();

Create queue targeting a pre-configured classes of devices:	queue(cpu_selector{});
	queue(gpu_selector{});
	queue(intel::fpga_selector{});
	queue(accelerator_selector{});
	queue(host_selector{});

Create queue targeting specific device (custom criteria):	class custom_selector : public device_selector {
	  int operator()(…… **// Any logic you want!**
	…
	queue(custom_selector{});

Always available

Kernel

- The kernel class encapsulates methods and data for executing code on the device when a command group is instantiated
- Kernel object is not explicitly constructed by the user
- Kernel object is constructed when a kernel dispatch function, such as parallel_for, is called

```
q.submit([&](handler& h) {
  h.parallel_for(range<1>(N), [=](id<1> i) {
    A[i] = B[i] + C[i]);
  });
});
```

DPC++ language and runtime

- DPC++ language and runtime consists of a set of C++ classes, templates, and libraries
- **Application scope** and **command group scope**:
  - Code that executes on the host
  - The full capabilities of C++ are available at application and command group scope
- **Kernel scope**:
  - Code that executes on the device.
  - At kernel scope there are limitations in accepted C++

Parallel Kernels

- Parallel Kernel allows multiple instances of an operation to execute in parallel.
- Useful to offload parallel execution of a basic for-loop in which each iteration is completely independent and in any order.
- Parallel kernels are expressed using the parallel_for function

for-loop in CPU application

```
for(int i=0; i < 1024; i++){
    a[i] = b[i] + c[i];
});
```

Offload to accelerator using **parallel_for**

```
h.parallel_for(range<1>(1024), [=](id<1> i){
    A[i] =  B[i] + C[i];
});
```

Basic Parallel Kernels

The functionality of basic parallel kernels is exposed via range, id and item classes

- range class is used to describe the iteration space of parallel execution
- id class is used to index an individual instance of a kernel in a parallel execution
- item class represents an individual instance of a kernel function, exposes additional functions to query properties of the execution range

```
h.parallel_for(range<1>(1024), [=](id<1> idx){
        // CODE THAT RUNS ON DEVICE
});
```

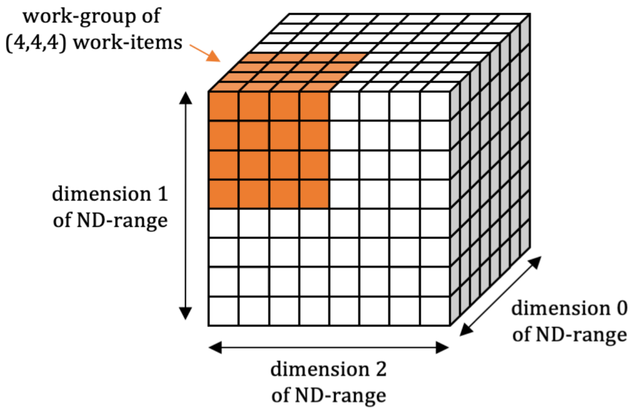
```
h.parallel_for(range<1>(1024), [=](item<1> item){
        auto idx = item.get_id();
        auto R = item.get_range();
        // CODE THAT RUNS ON DEVICE
});
```

ND-Range Kernels

Basic Parallel Kernels are easy way to parallelize a for-loop but does not allow performance optimization at hardware level.

ND-Range kernel is another way to expresses parallelism which enable low level performance tuning by providing access to local memory and mapping executions to compute units on hardware.

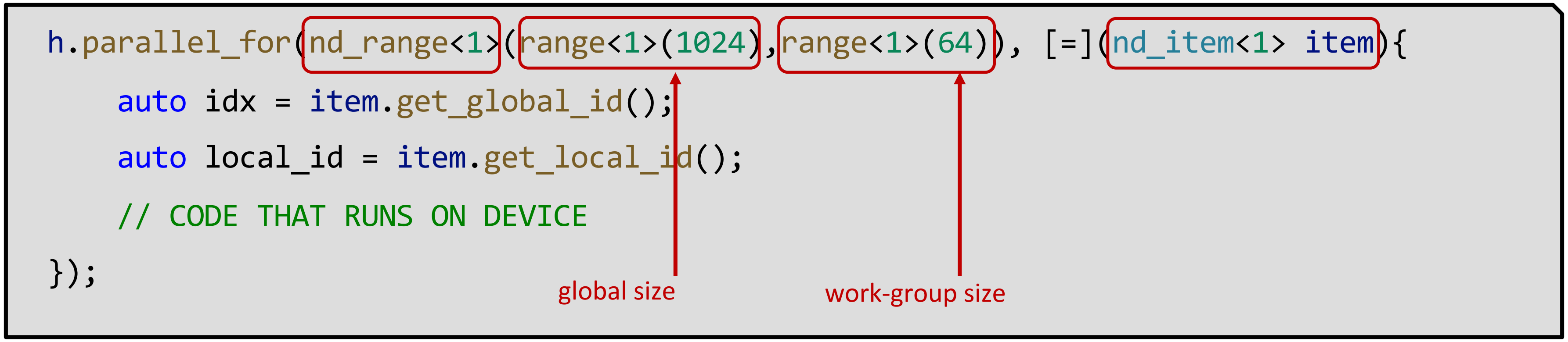
- The entire iteration space is divided into smaller groups called work-groups, work-items within a work-group are scheduled on a single compute unit on hardware.
- The grouping of kernel executions into work-groups will allow control of resource usage and load balance work distribution.



ND-Range

ND-Range Kernels

The functionality of nd_range kernels is exposed via nd_range and nd_item classes



```
h.parallel_for(nd_range<1>(range<1>(1024),range<1>(64)), [=](nd_item<1> item){
    auto idx = item.get_global_id();
    auto local_id = item.get_local_id();
    // CODE THAT RUNS ON DEVICE
});
```

global size
work-group size

- nd_range class represents a grouped execution range using global execution range and the local execution range of each work-group.
- nd_item class represents an individual instance of a kernel function and allows to query for work-group range and index.

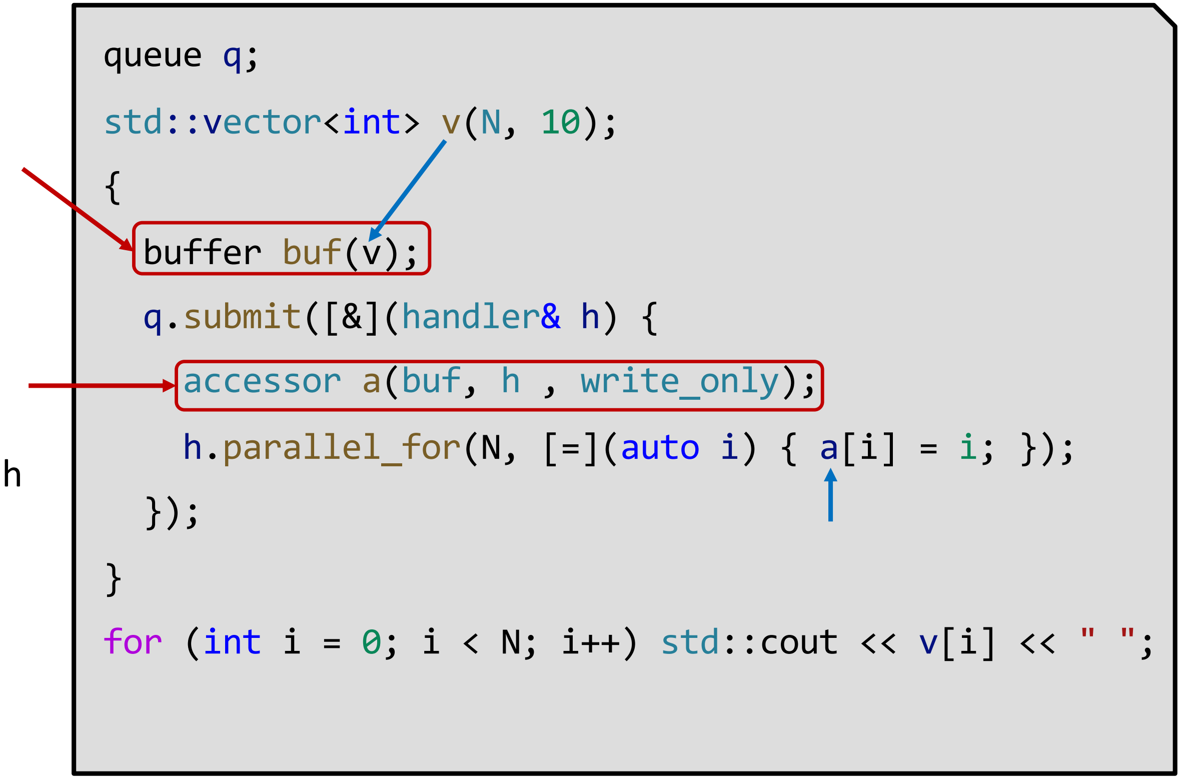
Buffer Memory Model

Buffers: Encapsulate data in a SYCL application

- Across both devices and host!

Accessors: Mechanism to access buffer data

- Create data dependencies in the SYCL graph that order kernel executions



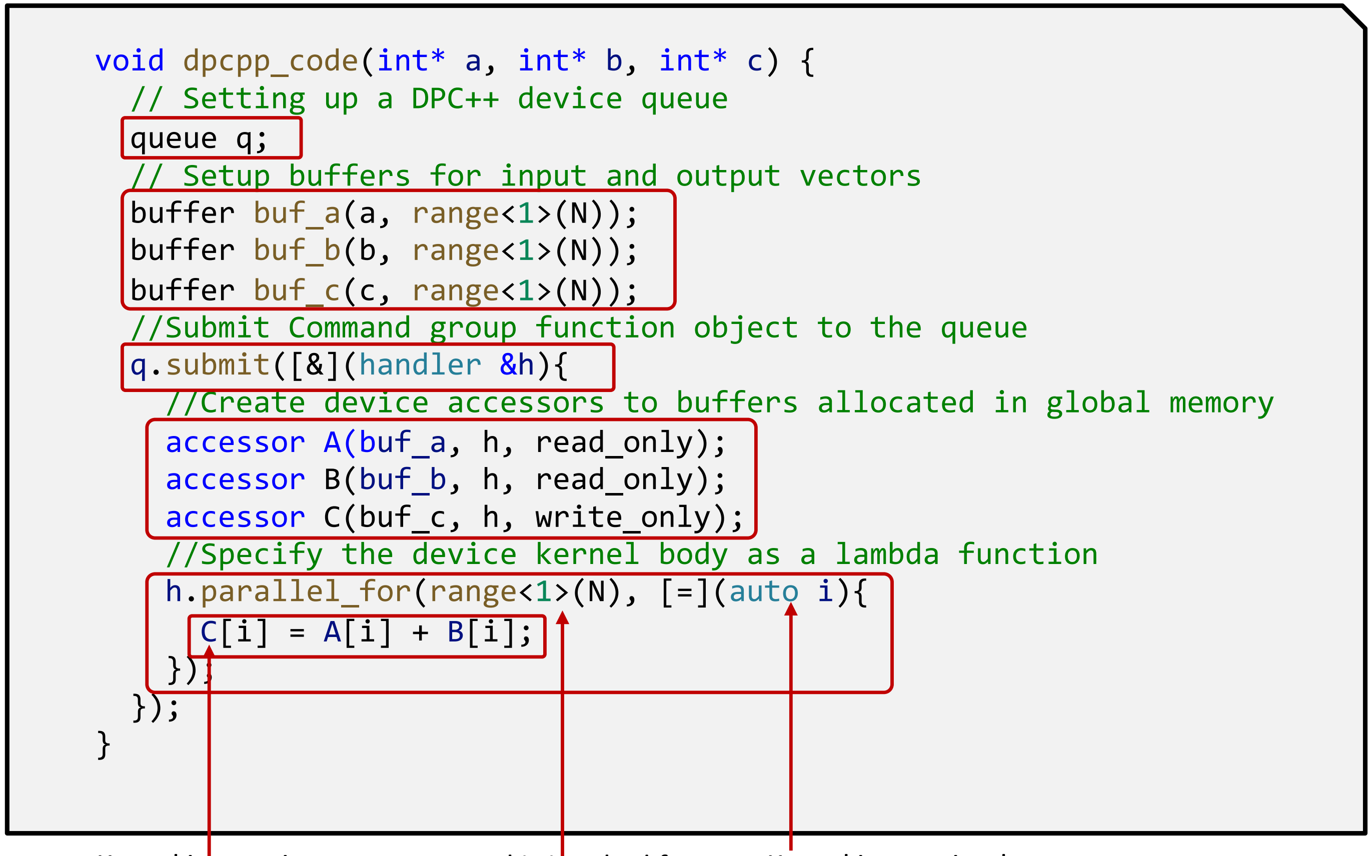
```
queue q;
std::vector<int> v(N, 10);
{
 buffer buf(v);
  q.submit([&](handler& h) {
   accessor a(buf, h , write_only);
    h.parallel_for(N, [=](auto i) { a[i] = i; });
  });
}
for (int i = 0; i < N; i++) std::cout << v[i] << " ";
```

DPC++ Code Anatomy

- oneAPI programs require the include of CL/sycl.hpp
- It is recommended to employ the namespace statement to save typing repeated references into the sycl namespace

```
#include <CL/sycl.hpp>
using namespace sycl;
```

DPC++ Code Anatomy



```
void dpcpp_code(int* a, int* b, int* c) {
  // Setting up a DPC++ device queue
  queue q;
  // Setup buffers for input and output vectors
  buffer buf_a(a, range<1>(N));
  buffer buf_b(b, range<1>(N));
  buffer buf_c(c, range<1>(N));
  //Submit Command group function object to the queue
  q.submit([&](handler &h){
    //Create device accessors to buffers allocated in global memory
    accessor A(buf_a, h, read_only);
    accessor B(buf_b, h, read_only);
    accessor C(buf_c, h, write_only);
    //Specify the device kernel body as a lambda function
    h.parallel_for(range<1>(N), [=](auto i){
      C[i] = A[i] + B[i];
    });
  });
}
```

Step 1: create a device queue
(developer can specify a device type via
device selector or use default selector)

Step 2: create buffers
(represent both host and
device memory)

Step 3: submit a command for (asynchronous)
execution

Step 4: create buffer accessors to
access buffer data on the device

Step 5: send a kernel (lambda) for
execution

Step 6: write a kernel

Kernel invocations
are executed in
parallel

Kernel is invoked for
each element of the
range

Kernel invocation has
access to the
invocation id

Done!
The results are copied to vector `c` at `buf_c` buffer destruction

Custom Device Selector

- The following code shows derived **device_selector** that employs a device selector heuristic. The selected device prioritizes a GPU device because the integer rating returned is higher than for CPU or other accelerator.

```
#include <CL/sycl.hpp>
using namespace cl::sycl;
class my_device_selector : public device_selector {
public:
  int operator()(const device& dev) const override {
    int rating = 0;
    if (dev.is_gpu() & (dev.get_info<info::device::name>().find("Intel") != std::string::npos))
      rating = 3;
    else if (dev.is_gpu()) rating = 2;
    else if (dev.is_cpu()) rating = 1;
    return rating;
  };
};
int main() {
  my_device_selector selector;
  queue q(selector);
  std::cout << "Device: “ << q.get_device().get_info<info::device::name>() << std::endl;
  return 0;
}
```

Hands-on Coding on Intel DevCloud

Intel VTune Profiler: DPC++ Profiling - Tune for CPU, GPU & FPGA

Analyze Data Parallel C++ (DPC++)

See the lines of DPC++ that consume the most time

Tune for Intel CPUs, GPUs & FPGAs

Optimize for any supported hardware accelerator

Optimize Offload

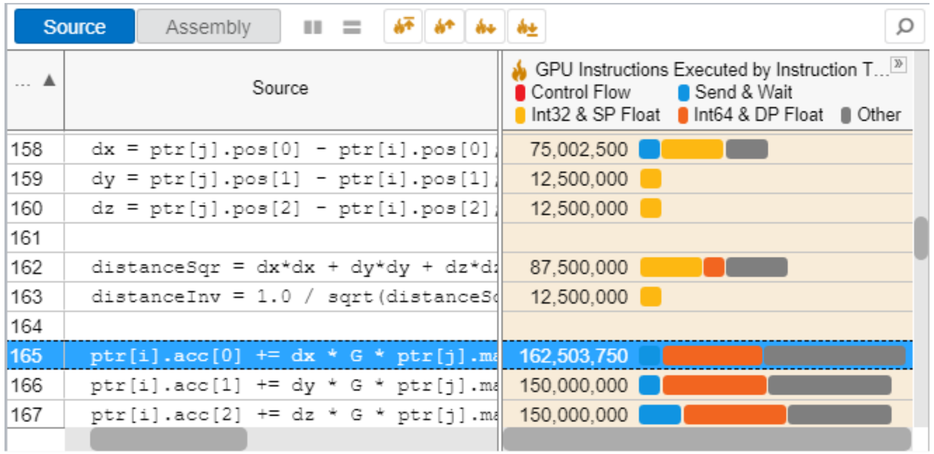
Tune OpenMP offload performance

Wide Range of Performance Profiles

CPU, GPU, FPGA, threading, memory, cache, storage…

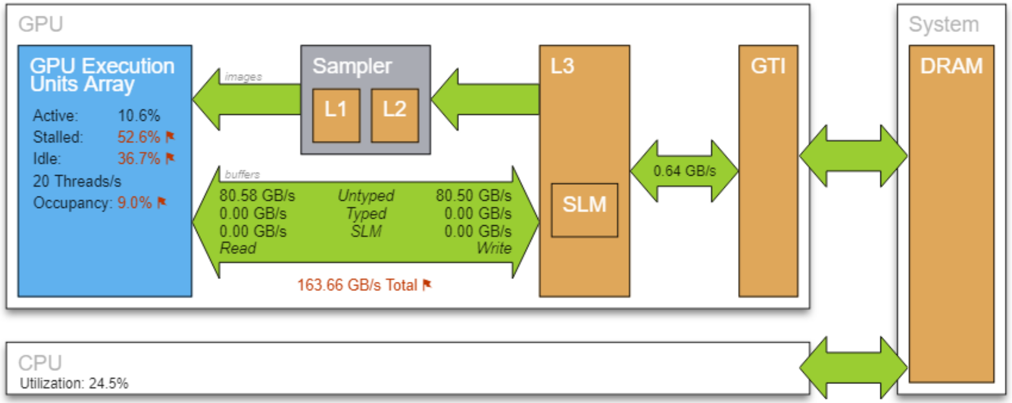
Supports Popular Languages

DPC++, C, C++, Fortran, Python, Go, Java, or a mix



GPU Instructions Executed by Instruction T…: 🟥 Control Flow 🟦 Send & Wait 🟨 Int32 & SP Float 🟧 Int64 & DP Float ■ Other

Source	GPU Instructions Executed by Instruction T…
`dx = ptr[j].pos[0] - ptr[i].pos[0];`	75,002,500
`dy = ptr[j].pos[1] - ptr[i].pos[1];`	12,500,000
`dz = ptr[j].pos[2] - ptr[i].pos[2];`	12,500,000
`distanceSqr = dx*dx + dy*dy + dz*d`	87,500,000
`distanceInv = 1.0 / sqrt(distanceS`	12,500,000
`ptr[i].acc[0] += dx * G * ptr[j].ma`	162,503,750
`ptr[i].acc[1] += dy * G * ptr[j].ma`	150,000,000
`ptr[i].acc[2] += dz * G * ptr[j].ma`	150,000,000



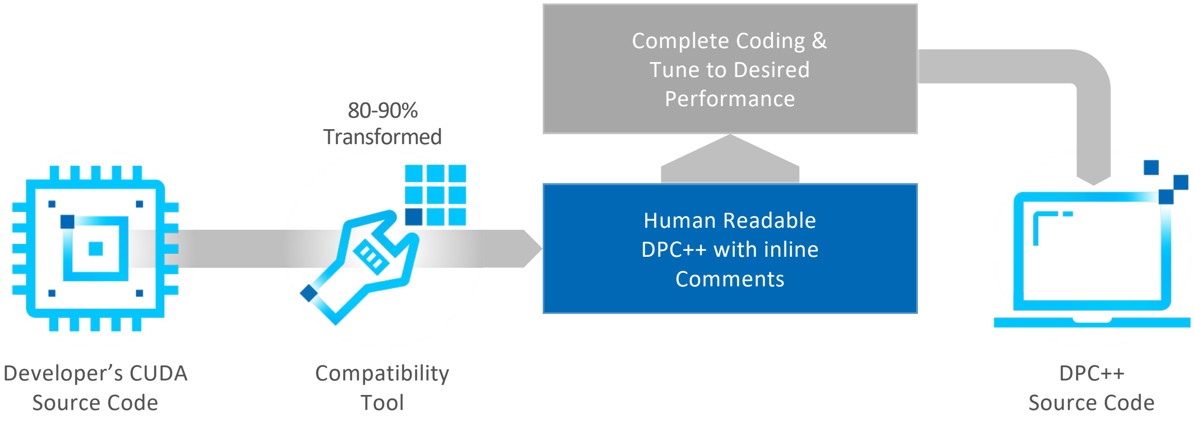
There will still be a need to tune for each architecture.

Hands-on Intel VTune Profiler on Intel DevCloud

Intel DPC++ Compatibility Tool: Minimizes Code Migration Time

- Assists developers migrating code written in CUDA to DPC++ once, generating **human readable** code wherever possible
- ~80-90% of code typically migrates automatically
- Inline comments are provided to help developers finish porting the application

Intel DPC ++ Compatibility Tool Usage Flow


Complete Coding & Tune to Desired Performance
80-90% Transformed
Human Readable DPC++ with inline Comments
Developer's CUDA Source Code
Compatibility Tool
DPC++ Source Code

Hands-on Intel DPC++ Compatibility Tool on Intel DevCloud

Thanks!