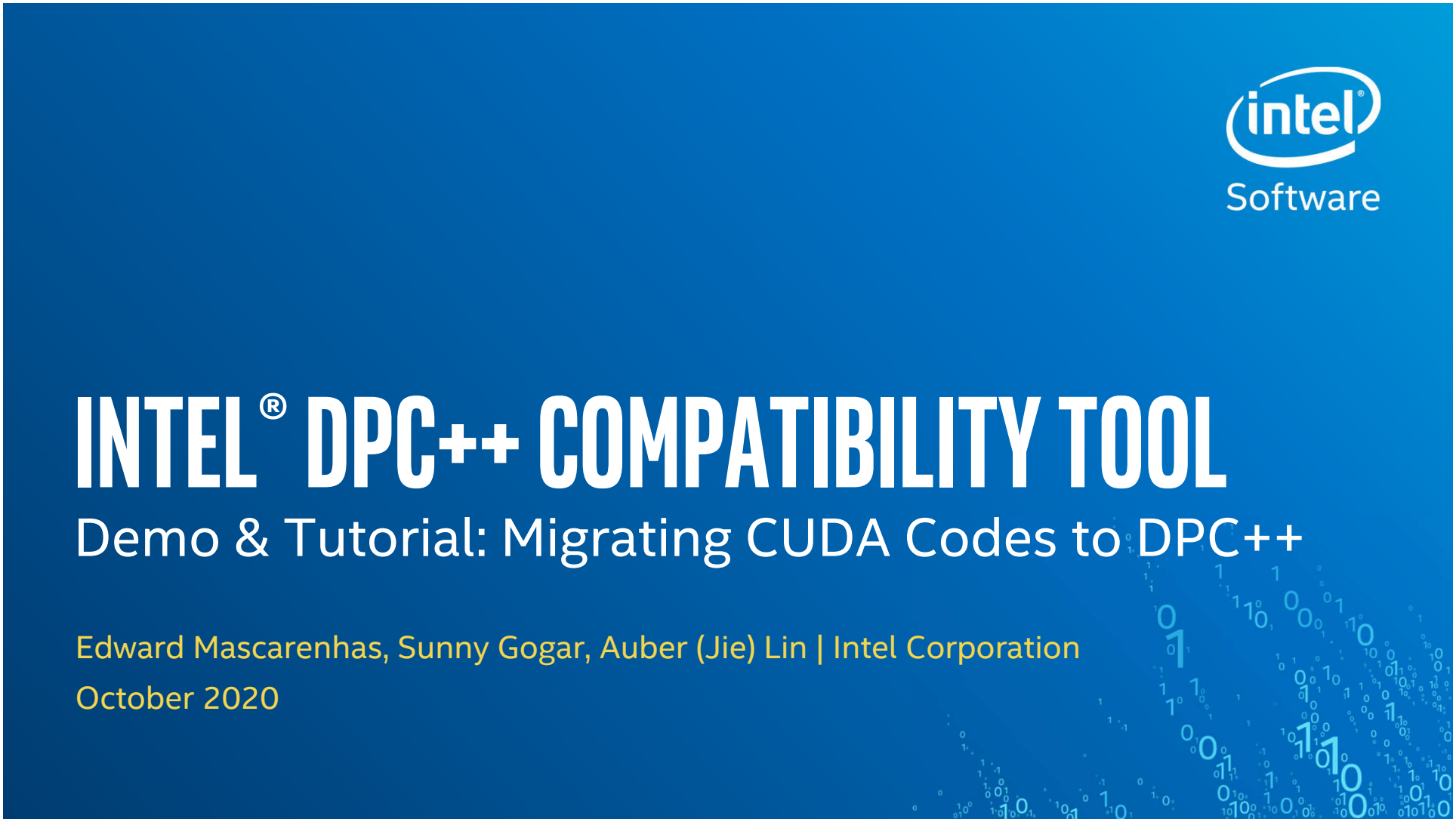

intel
Software
INTEL® DPC++ COMPATIBILITY TOOL
Demo & Tutorial: Migrating CUDA Codes to DPC++
Edward Mascarenhas, Sunny Gogar, Auber (Jie) Lin | Intel Corporation
October 2020

NOTICES AND DISCLAIMERS

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

All product plans and roadmaps are subject to change without notice.

Intel technologies may require enabled hardware, software or service activation.

Results have been estimated or simulated.

No product or component can be absolutely secure.

Intel does not control or audit third-party data. You should consult other sources to evaluate accuracy.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Agenda

- oneAPI Brief Overview
- Intel® DPC++ Compatibility Tool Workflow
- Migration Flow and Vector-Add Example
- Demo Tutorial
  - Migrate Single CUDA* File Project
  - Migrate Multi CUDA Files Project
- Code Review and Best Known Methods for Migration
- Eclipse and Visual Studio Integration
- Key Takeaways

oneAPI Core Concept

Project oneAPI delivers a unified programming model to simplify development across diverse architectures

oneAPI has two main parts:	
oneAPI Industry Initiative	**oneAPI Intel Product**

Common developer experience across Scalar, Vector, Matrix, and Spatial (SVMS) architectures

Unified and simplified language and libraries for expressing parallelism

Uncompromised native high-level language performance

Support for CPU, GPU, AI, and FPGA

Based on industry standards and open specifications

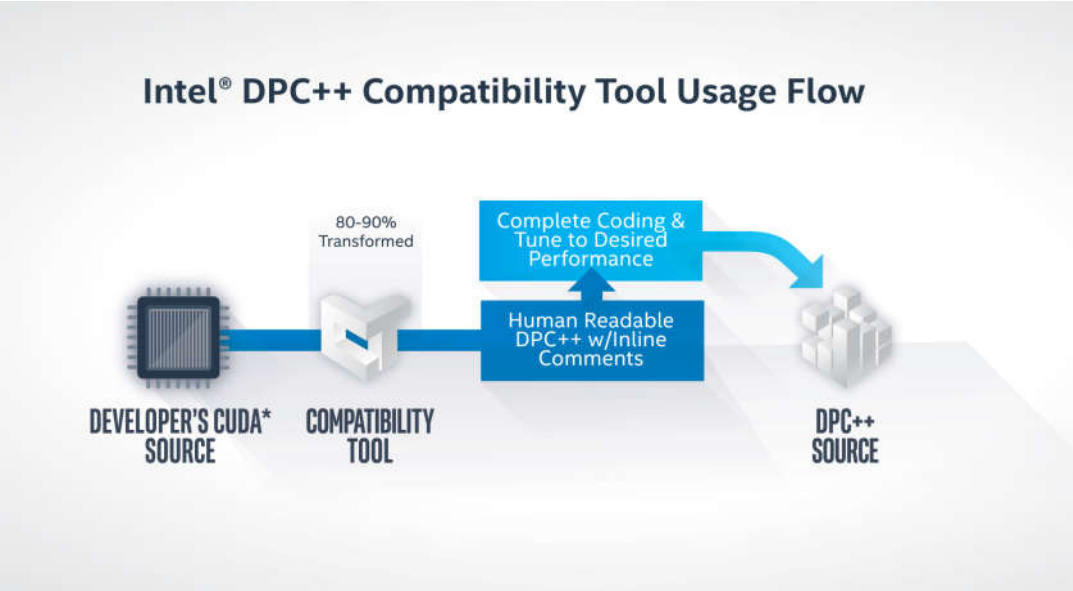
Intel® DPC++ Compatibility Tool

Minimizes Code-Migration Time

Assists developers migrating code written in CUDA to DPC++ by generating DPC++ code wherever possible

Expect up to 80-90% of code to migrate automatically

Inline comments are provided to help developer complete code



MIGRATING VECTOR ADD EXAMPLE

Vector-Add Example: Migration with Intel® DPC++ Compatibility Tool

CUDA

```
#include <cuda.h>
#include <stdio.h>
#define VECTOR_SIZE 256
```

```
__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;

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

DPC++

```
#include <CL/sycl.hpp>
#include <dpct/dpct.hpp>
#define VECTOR_SIZE 256
```

1

```
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)];
}
```

2

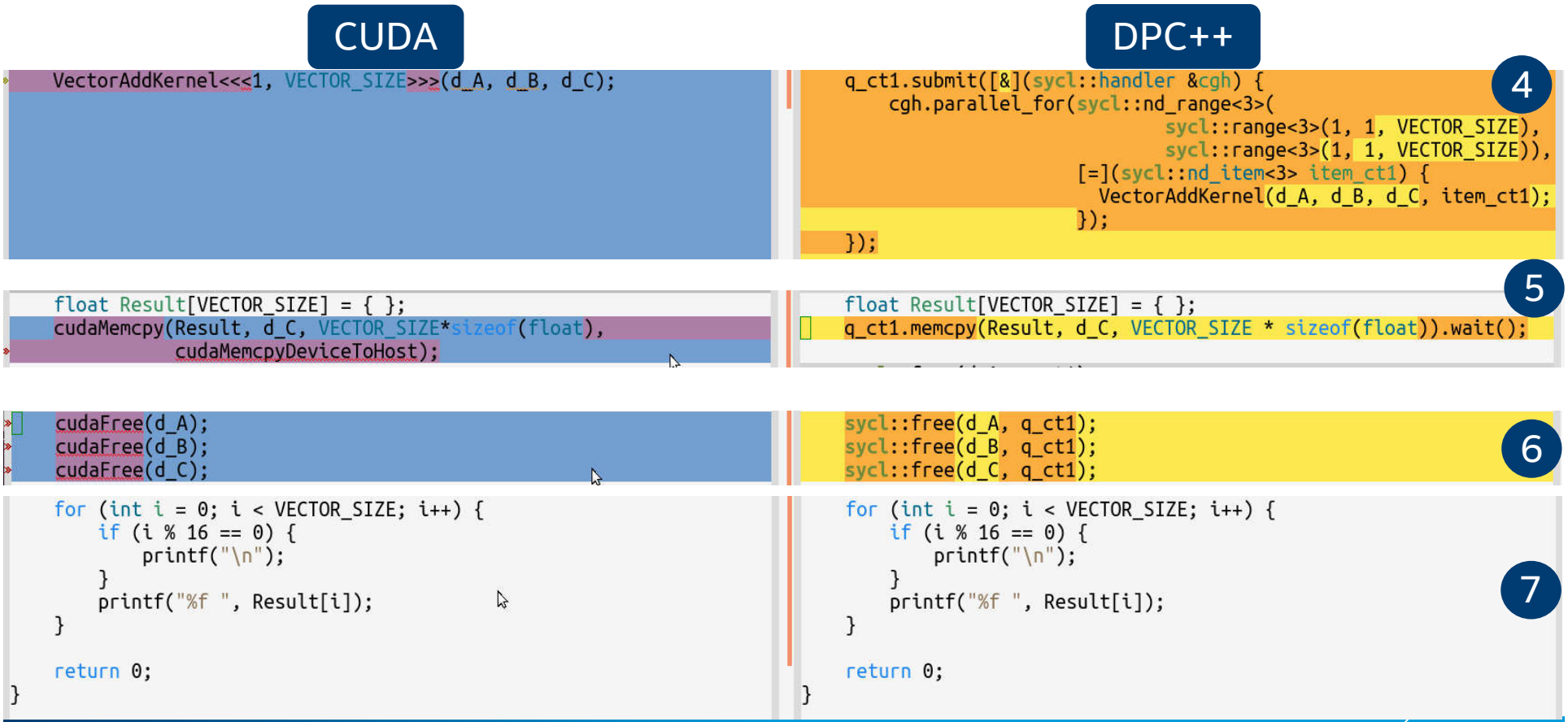
```
int main()
{
    dpct::device_ext &dev_ct1 = dpct::get_current_device();
    sycl::queue &q_ct1 = dev_ct1.default_queue();
    float *d_A, *d_B, *d_C;

    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);
```

3

https://software.intel.com/en-us/download/sample-vector-add

Vector-Add Migration Example (continued)

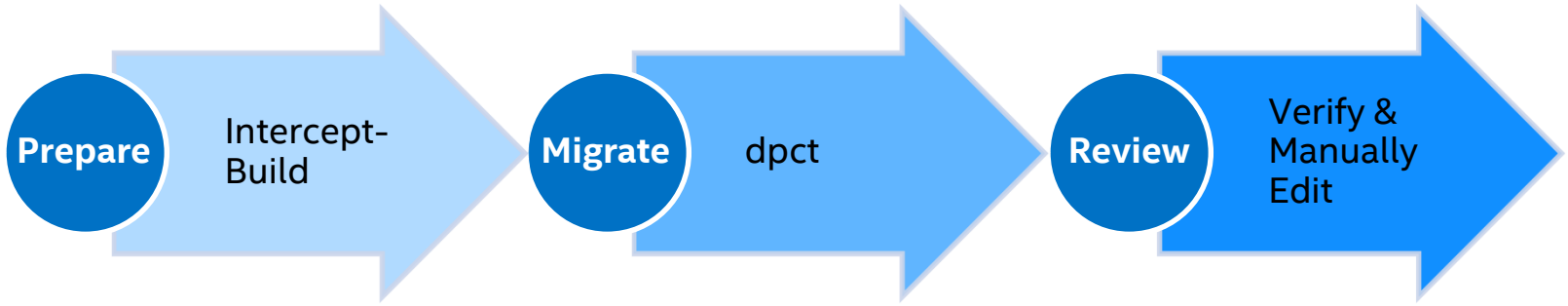


https://software.intel.com/en-us/download/sample-vector-add

MIGRATING NEEDLEMAN WUNSCH AND HYDROC EXAMPLES

Migration Flow

Typical preparation steps for simple to complex projects



Demo: Single CUDA File Project Migration

- Rodinia Benchmark Suite v3.1 – Introduction
- Setting/Verifying the Environment for Intel® DPC++ Compatibility Tool
- Demo
  - Planning for Migration
  - Compatibility Tool Options
  - Migrating Needleman Wunsch Application

http://rodinia.cs.virginia.edu/doku.php
https://software.intel.com/en-us/get-started-with-intel-dpcpp-compatibility-tool
https://software.intel.com/en-us/intel-dpcpp-compatibility-tool-user-guide-usage-workflow-overview

Demo: HydroC - Multi CUDA Files Project Migration

- Setting/Verifying the Environment for Intel® DPC++ Compatibility Tool
- Demo
  - Planning for Migration; Understanding the Application File …
  - *intercept-build* Options
  - Compatibility Tool Options
  - Migrating HydroC Application

https://github.com/HydroBench/Hydro/tree/master/HydroC/cuHydroC_2DMpi/Src
https://github.com/HydroBench/Hydro
https://github.com/HydroBench/Hydro/blob/master/License.txt

Code Review or Rewrite Needed

Diagnostic Reference

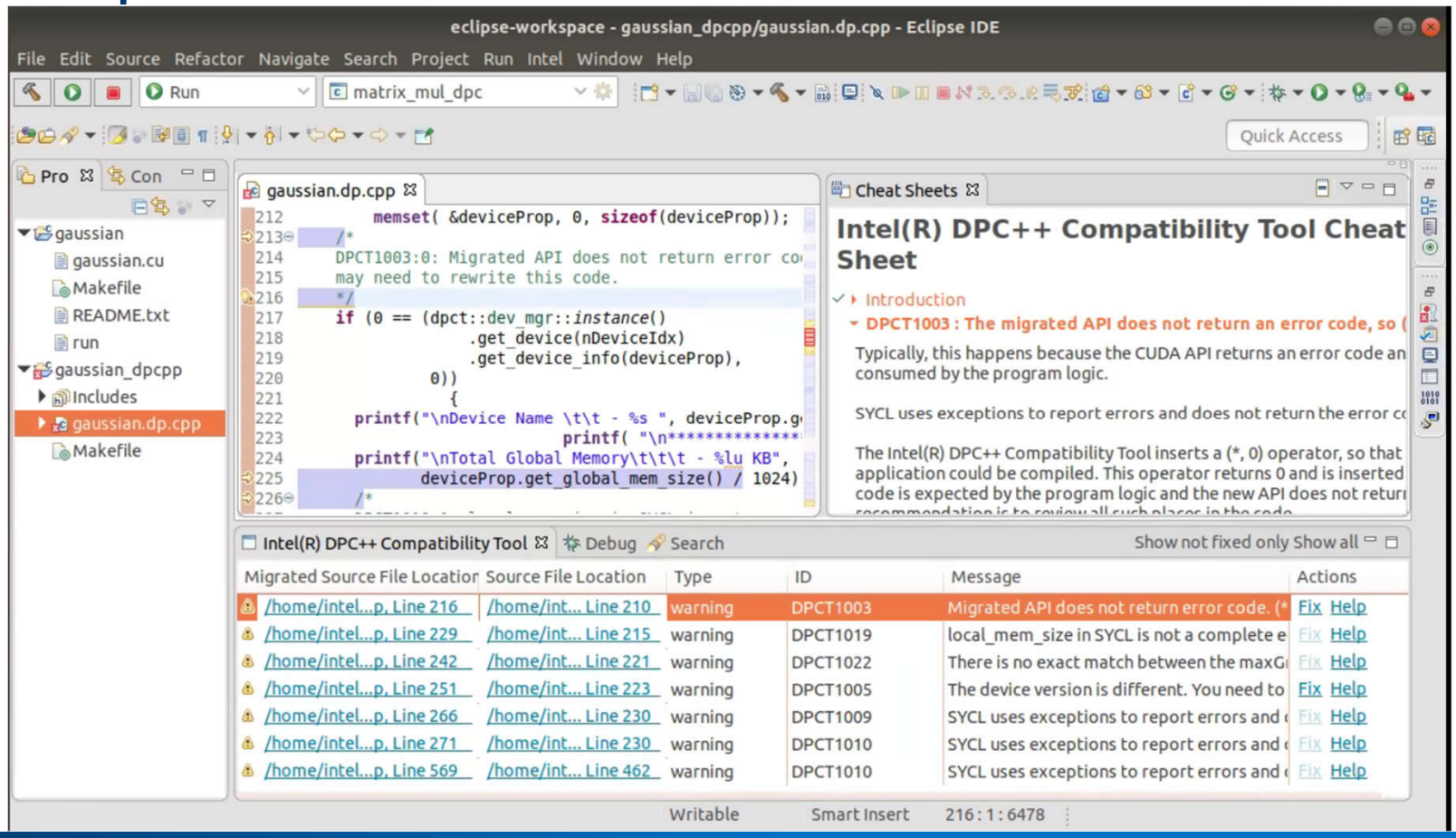
- Error code logic replaced with (*,0) code or commented out
- Equivalent DPC++ API not available
- CUDA Compute Capability-dependent logic
- Hardware-dependent API (clock())
- Missing features (Unsupported API)
- Execution time measurement logic
- Handling built-in vector type conflicts
- Migration of cuBLAS API (Review arguments list)

General Best Known Methods (BKMs)

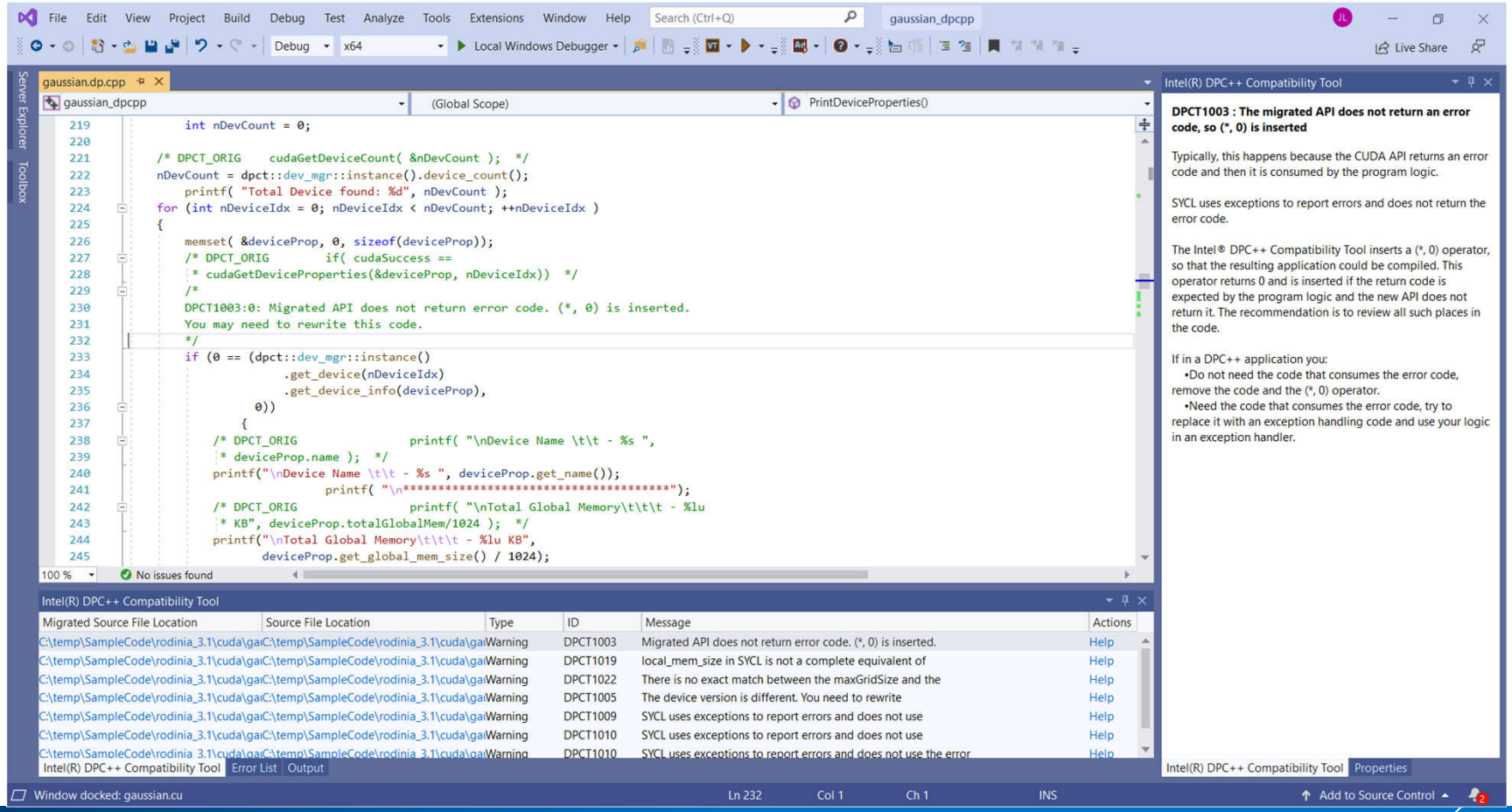
- Migrate Incrementally
  - If you see *dpct* generate multiple errors when migrating a long list of CUDA source files in one run, do it one-by-one
- Start with a clean project - “make clean” before running “intercept-build make”

USING PLUGINS WITH IDE

Eclipse: Gaussian



Visual Studio 2019: Gaussian



Key Takeaways

- OneAPI delivers a unified programming model to simplify development across diverse architectures
- Intel DPC++ Compatibility tool assists developers in migrating code written in CUDA to DPC++, increasing developer productivity
- DPC++ is an open specification for a portable, architecture-neutral language for expressing parallelism; it is based on industry standards

Are You Ready to Try oneAPI?

1. Identify potential workloads/candidates for testing
   a. Download DPCT and migrate code to DPC++ on-prem, if applicable
   b. Test, tune and optimize your code or test samples in the Intel® DevCloud—a cloud-based development sandbox environment that gives you full access to the latest Intel® hardware and oneAPI software https://software.intel.com/devcloud/oneapi
2. Learn more at http://software.intel.com/oneapi the channel to documentation, downloads, access to Intel® Devcloud, and access to support forum

QUESTIONS?

THANK YOU