INTEL® DPC++ COMPATIBILITY TOOL

AGENDA

- What is Intel® DPC++ Compatibility Tool?
- Migration Flow
  - One File Migration
  - vecAdd Example
  - Migration of Larger Code Bases
- Advanced Options

INTEL® DPC++ COMPATIBILITY TOOL(BETA)
MINIMIZES CODE MIGRATION TIME

- Intel® oneAPI Base Toolkit component
- Assists developers migrating code written in CUDA* to DPC++, generating **human readable** code wherever possible
- Inline comments are provided to help developers complete their code
- Windows* and Linux*, IDE integration support

https://software.intel.com/en-us/get-started-with-intel-dpcpp-compatibility-tool

BEFORE YOU BEGIN

Setup the environment

Linux: *source /Path/to/dpcpp-ct/latest/env/vars.sh*

or

source /opt/intel/inteloneapi/setvars.sh

Windows: *Drive:\Path\to\dpcpp-ct\latest\env\vars.bat*

or

C:\Program Files (x86)\inteloneapi\setvars.bat

Command Line Syntax

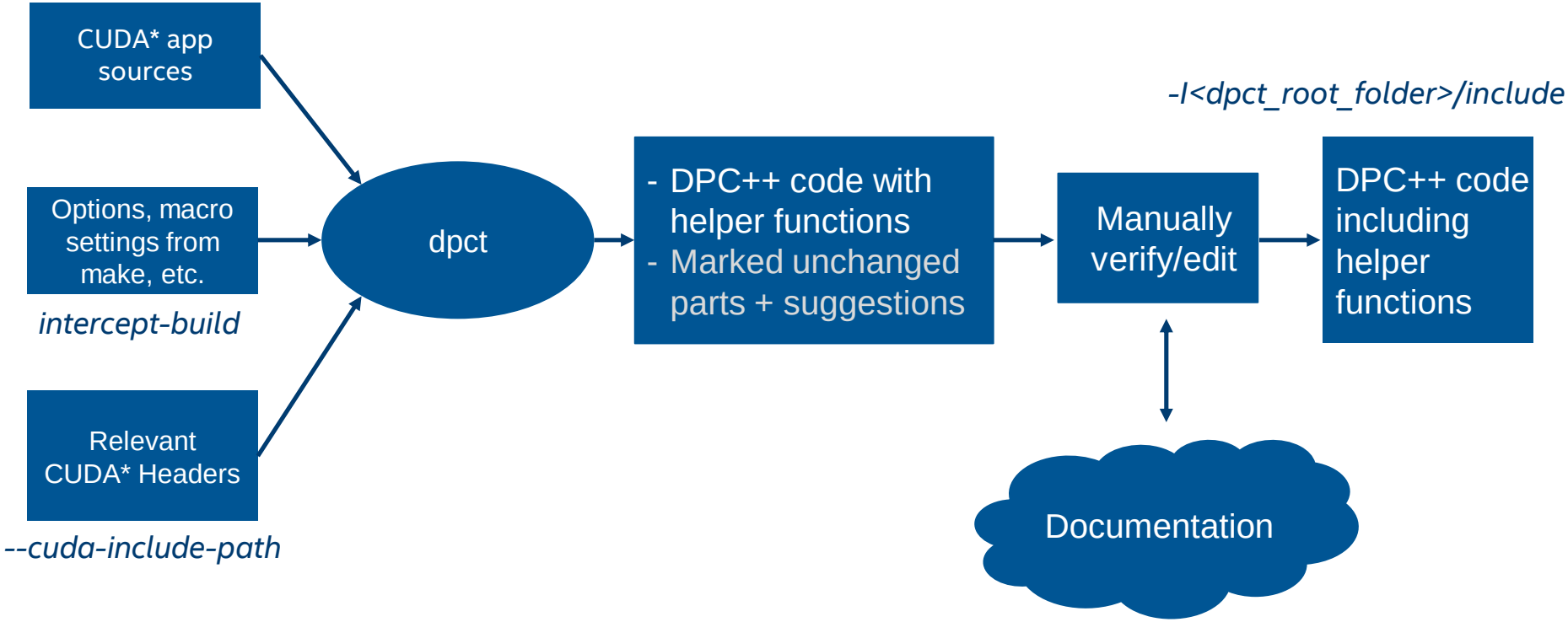
dpct [options] [<source0>... <sourceN>]

Built-in Usage Information

dpct --help

dpct -- -help

USAGE WORKFLOW



https://software.intel.com/en-us/get-started-with-intel-dpcpp-compatibility-tool-diagnostics-reference

WHAT IS NOT MIGRATED?

Tool highlights issues with migration via warnings and code comments

- grep for *DPCT10*

Warnings format:

```
/path/to/file.hpp:26:1: warning: DPCT10XX:0: text of the warning.
// source code line for which warning was generated
^
```

Check documentation for detailed explanation and suggestions to fix

https://software.intel.com/en-us/get-started-with-intel-dpcpp-compatibility-tool-diagnostics-reference

SIMPLE ONE FILE MIGRATION

dpct vectorAdd.cu

- Requires certain CUDA* header files. Default files location:
  - */usr/local/cuda/include*
  - */usr/local/cuda-x.y/include*
- Supported versions: 8.0, 9.0, 9.1, 9.2, 10.0, 10.1
- Migrates CUDA* code *vectorAdd.cu* to DPC++ code *dpct_output/vectoradd.dp.cpp*

dpct --cuda-include-path=<path/to/cuda/include> vectorAdd.cu

dpct --extra-arg="-std=c++11" code_wtih_c++11.cu

```
#include <cuda.h>

#define VECTOR_SIZE 4
__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));



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



  float Result[VECTOR_SIZE] = { };
  cudaMemcpy(Result, d_C,VECTOR_SIZE*sizeof(float),cudaMemcpyDeviceToHost);

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

Header files

Kernel

Mem alloc

Kernel call

Mem copy

Mem free

```
#include <CL/sycl.hpp>
#include <dpct/dpct.hpp>
#define VECTOR_SIZE 4
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)];
}

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);

  q_ct1.submit([&](sycl::handler &cgh){
    cgh.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);
    });
  });

  float Result[VECTOR_SIZE] = { };
  q_ct1.memcpy(Result, d_C, VECTOR_SIZE * sizeof(float)).wait();

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

MIGRATION OF LARGER CODE BASES

1. Create a compilation database file

   intercept-build make

2. Migrate your source to DPC++

   dpct -p compile_commands.json -in-root=$PROJ_DIR -out-root=dpcpp_out *.cu

3. Verify the source for correctness and fix not migrated parts

https://software.intel.com/en-us/intel-dpcpp-compatibility-tool-user-guide-usage-workflow-overview

ADVANCED OPTIONS

--enable-ctad

- Use a C++17 class template argument deduction (CTAD) in generated code

  Default: off

```
dpct::get_default_queue_wait().submit([&](sycl::handler &cgh) {
  cgh.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, 1) *
                                     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);
                   });
});
```

default

```
dpct::get_default_queue_wait().submit([&](sycl::handler &cgh) {
  cgh.parallel_for(
      sycl::nd_range(sycl::range(1, 1, 1) * 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);
      });
});
```

--enable-ctad

https://github.com/intel/llvm/tree/sycl/sycl/doc/extensions/deduction_guides

ADVANCED OPTIONS

--sycl-named-lambda

- Generates kernels with the kernel name

  Default: off

```
dpct::get_default_queue_wait().submit([&](sycl::handler &cgh) {
  cgh.parallel_for(sycl::nd_range<3>(sycl::range<3>(1, 1, 1) *
                                     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);
                   });
});
```

default

```
dpct::get_default_queue_wait().submit([&](sycl::handler &cgh) {
  cgh.parallel_for<dpct_kernel_name<class VectorAddKernel_5218d9>>(
      sycl::nd_range<3>(sycl::range<3>(1, 1, 1) *
                        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);
      });
});
```

--sycl-named-lambda

https://github.com/intel/llvm/tree/sycl/sycl/doc/extensions/UnnamedKernelLambda

ADVANCED OPTIONS

--no-cl-namespace-inline

- Do not use cl namespace (cl::) inlining

  Default: off

```
d_A = (float *)sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                   dpct::get_current_device(),
                                   dpct::get_default_context());
d_B = (float *)sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                   dpct::get_current_device(),
                                   dpct::get_default_context());
d_C = (float *)sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                   dpct::get_current_device(),
                                   dpct::get_default_context());
```

default

```
d_A = (float *)cl::sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                       dpct::get_current_device(),
                                       dpct::get_default_context());
d_B = (float *)cl::sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                       dpct::get_current_device(),
                                       dpct::get_default_context());
d_C = (float *)cl::sycl::malloc_device(VECTOR_SIZE * sizeof(float),
                                       dpct::get_current_device(),
                                       dpct::get_default_context());
```

--no-cl-namespace-inline

NOTICES & DISCLAIMERS

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at intel.com, or from the OEM or retailer.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit www.intel.com/benchmarks.

INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS". NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Optimization Notice

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.
Notice revision #20110804

Q&A