Issues highlight during C-DAC migration from CUDA to SYCL

Yantao Zhang

Software Technical Consulting Engineer

Runtime Error: CL_INVALID_KERNEL_NAME

terminate called after throwing an instance of 'cl::sycl::runtime_error'

what(): No kernel named _ZTS16dpct_kernel_nameIJZZ12FDOperator_4iiENKUlRN2cl4sycl7handlerEE_clES3_E25cuda_fdoperator420_707dc5EE was found -46 (CL_INVALID_KERNEL_NAME)

Final Makefile

- Kernel wasn't properly included in the final binary
- Rule for link line:

  mpicxx -o $@ $^ $(LIB) -fsycl –lsycl

  - -fsycl : Enables a program to be compiled as a SYCL program rather than as plain C++11 program.
  - Some mpicxx build uses g++ as the underlying driver, may need option e.g. –cxx=icpx to switch driver

Accuracy issue

- CUDA

```
cudaStream_t stream1;
cudaStreamCreate(&stream1);
cudaMemsetAsync( … ,          ,stream1);
cuda_kernel<<< disg, dimb>>>(….);
cudaMemsetAsync(…,          ,stream1);
…
```

- SYCL

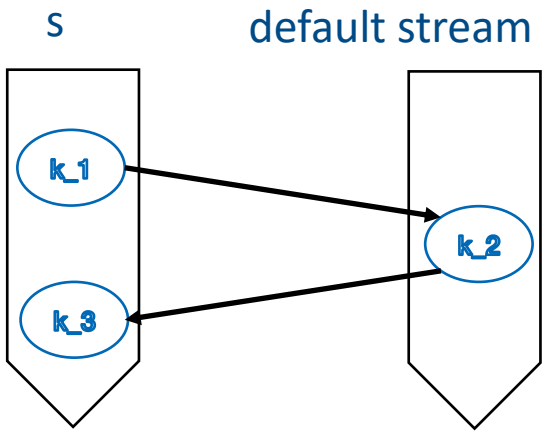
```
sycl::queue  * stream1 =
dpct::get_current_device().create_queue();
stream1->memset( … );
dpct::get_default_queue().submit(
    …  cuda_kernel  … );
stream1->memset( … );
…
```

Accuracy issue

- CUDA stream
  - The legacy default stream is an implicit stream which synchronizes with all other streams in the same CUcontext except for non-blocking streams (ref)
    - No operation in the default stream will begin until all previously issued operations *in any stream on the device* have completed, and an operation in the default stream must complete before any other operation (in any stream on the device) will begin (ref)
  - Operations submitted to CUDA stream execute in issue-order on the GPU
  - All 3 kernels will be serialized in the following example:

```
cudaStream_t s;
k_1<<<1, 1, 0, s>>>();
k_2<<<1, 1>>>();
k_3<<<1, 1, 0, s>>>();
```

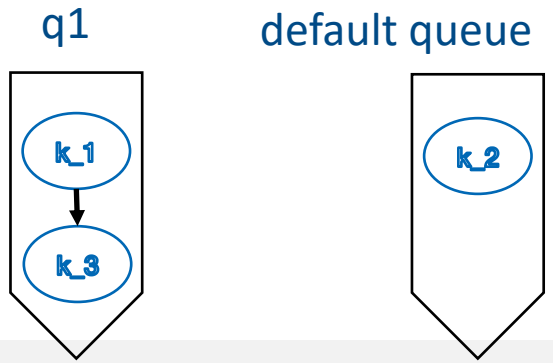
  - k_2 will block on k_1
  - k_3 will block on k_2



Accuracy issue

- SYCL queue
  - Actions submitted to sycl queues are asynchronous with respect to host and non-blocking.
    - Can be synchronized with host with wait(), or wait_and_throw()
  - SYCL queue created by dpct are in-order by default.
    - Regular sycl queue executes actions out of order (sycl::queue q;)
  - In the following example, kernel k_2 will be executed in parallel w.r.t. k_1 and k_3.

```
sycl::queue q1 = dpct::get_current_device().create_queue();
q1.submit(  …   k_1   … );
dpct::get_default_queue.submit (..  k_2  …);
q1.submit(  …   k_3   … );
```



Accuracy issue

- CUDA

```
cudaStream_t stream1;
cudaStreamCreate(&stream1);
cudaMemsetAsync( … ,          ,stream1);
cuda_kernel<<< dimg, dimb>>>(….);
cudaMemsetAsync(…,          ,stream1);
…
```

- SYCL

```
sycl::queue  * stream1 = &(dpct::get_default_queue());
stream1->memset( … );
dpct::get_default_queue().submit(
                …  cuda_kernel  … );
stream1->memset( … );
…
```

- DPCT default queue is created with in-order property