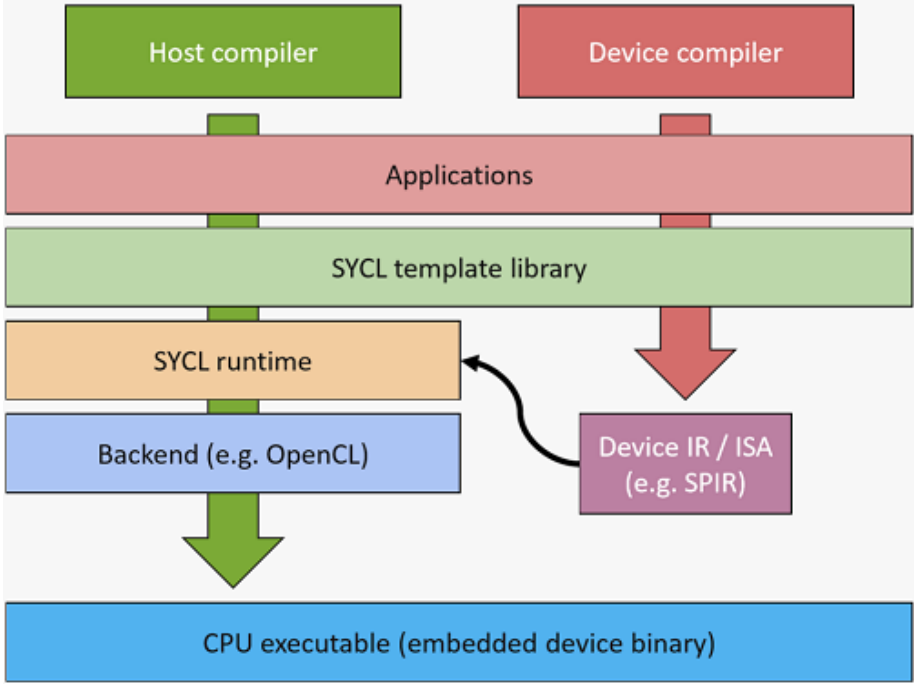
SUMMARY

1. WHAT IS SYCL?



- Single source, high-level, standard C++ programming model, that can target a range of heterogeneous platforms
- Provides high-level abstractions over common boilerplate code

2. ENQUEUEING A KERNEL

```
class my_kernel;

queue deviceQueue;
deviceQueue.submit([&](handler& cgh){

  auto os = sycl::stream(1024, 128, cgh);

  cgh.single_task<my_kernel>([=]() {
    os << "Hello world!\n";
  });
}).wait();
```

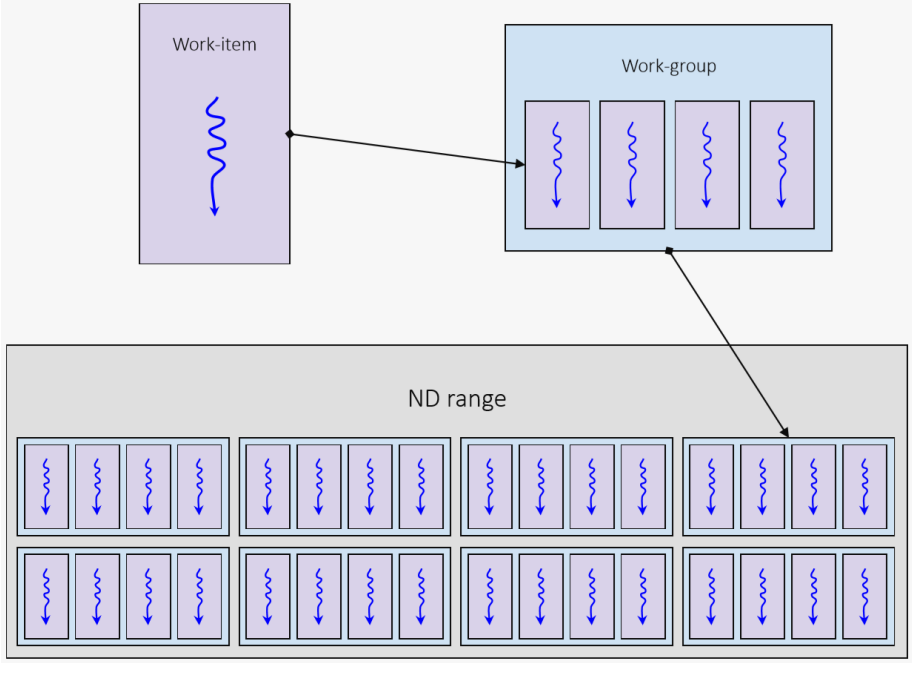
- Series of commands are enqueued via command groups
  - Performed via `sycl::queue::submit`
- Commands are scheduled for execution on a device
- Kernels submitted as kernel functions, either as C++ lambdas or function objects
- There are restrictions on kernel code because of device limitations
- Can use a `sycl::stream` to write text on device

3. MANAGING DATA

```
class my_kernel;
queue deviceQueue;

// Create a buffer and allocate USM memory
buffer<int> user_buffer{range{128}};
int* usm_ptr = malloc_device<int>(128, deviceQueue

deviceQueue.submit([&](handler& cgh){
  // Request access to buffer
  accessor user_acc{user_buffer, cgh, write_only};

  // USM pointer doesn't need to request access

  cgh.single_task<my_kernel>([=]() {
    user_acc[0] = 1;
    usm_ptr[0] = 2;
  });
}).wait();
```

- Two models for managing data: Buffer/accessor model and Unified Shared Memory
- SYCL separates the storage (buffer) and access of data (accessor)
- Different types of accessor provide different ways to access data
- In buffer/accessor model the scheduler takes care of data movement
- Access modes specify how to access data (read/write/no_init)
- USM memory can be allocated as host, device, or shared
- USM requires manual operations

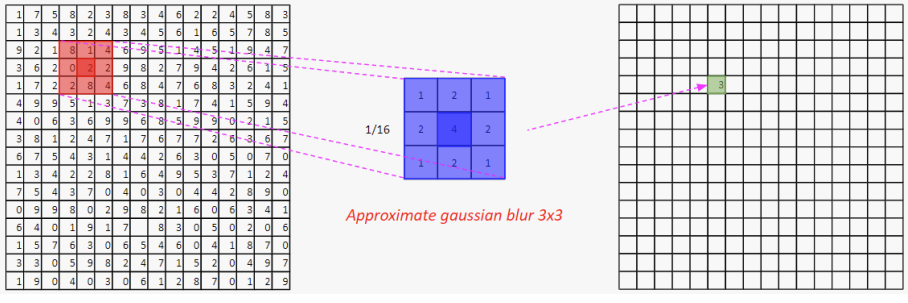
4. DATA PARALLELISM AND ND-RANGE KERNELS



- Task parallelism: Executes separate tasks simultaneously
- Data parallelism: Performs the same task on multiple data elements
- Work-items perform computation, execute independently (threads)
- Work-items grouped into work-groups. Work-groups invoked within an ND-range. Work-items within a group can synchronize
- Each work-item has private memory, can't access others'

5. IMAGE CONVOLUTION

$$G = h \otimes F \qquad G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v]F[i+u, j+v]$$



- Embarrassingly parallel algorithm: many independent calculations per work-item
- Each pixel multiplied with the filter