

# MULTIPLE DEVICES

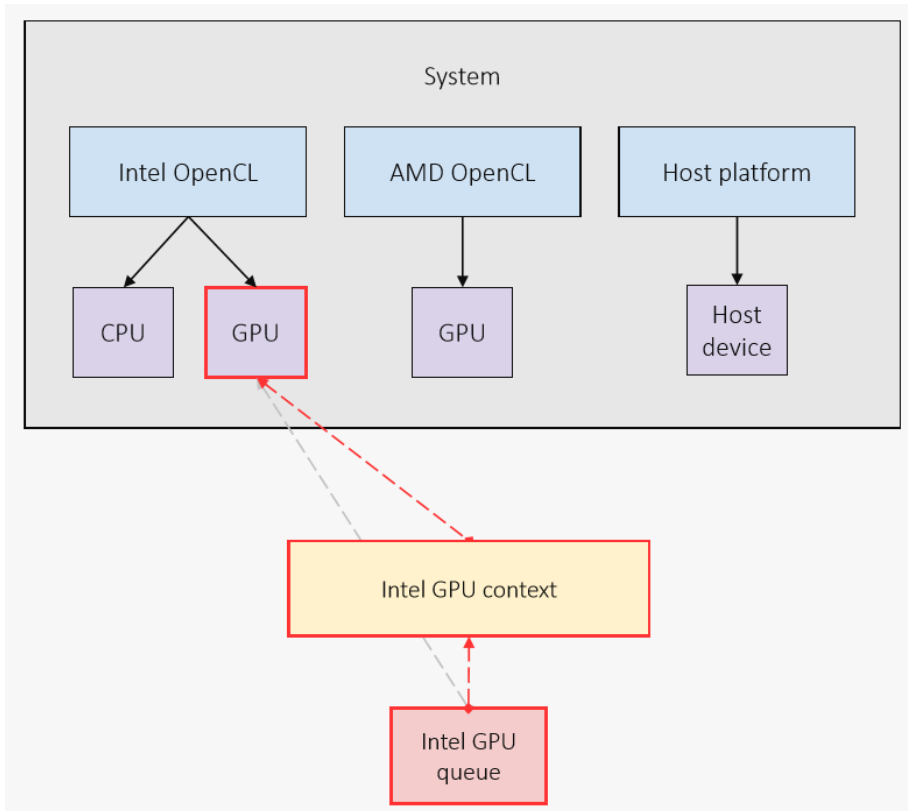
# LEARNING OBJECTIVES

- Learn about contexts and what they are used for
- Learn about how create dependencies across devices
- Learn about moving data between devices

# WHAT IS A CONTEXT

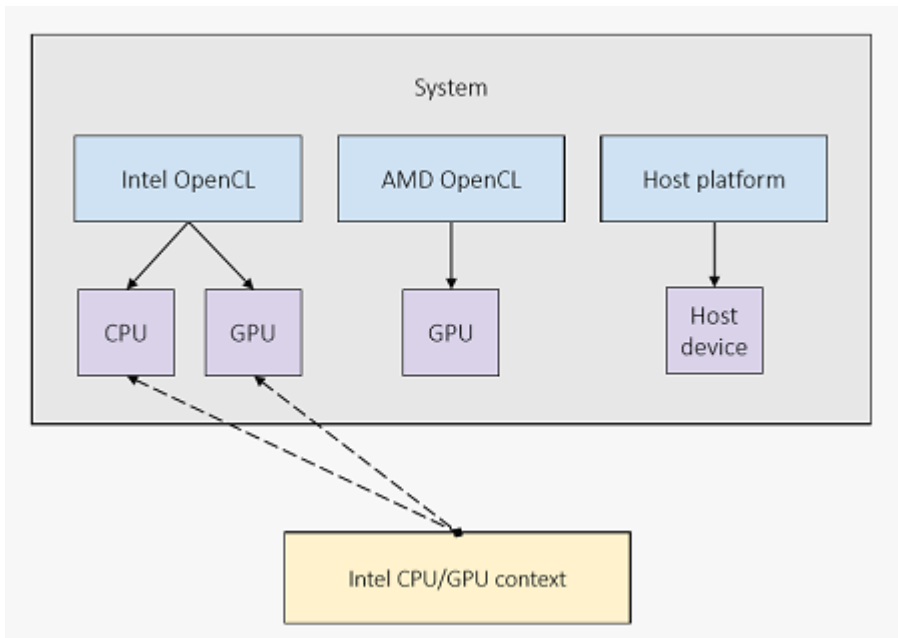
- In SYCL the underlying execution and memory resources of a platform and its devices is managed by creating a context
- A context represents one or more devices, but all devices must be associated with the same platform

# IMPLICIT CONTEXT



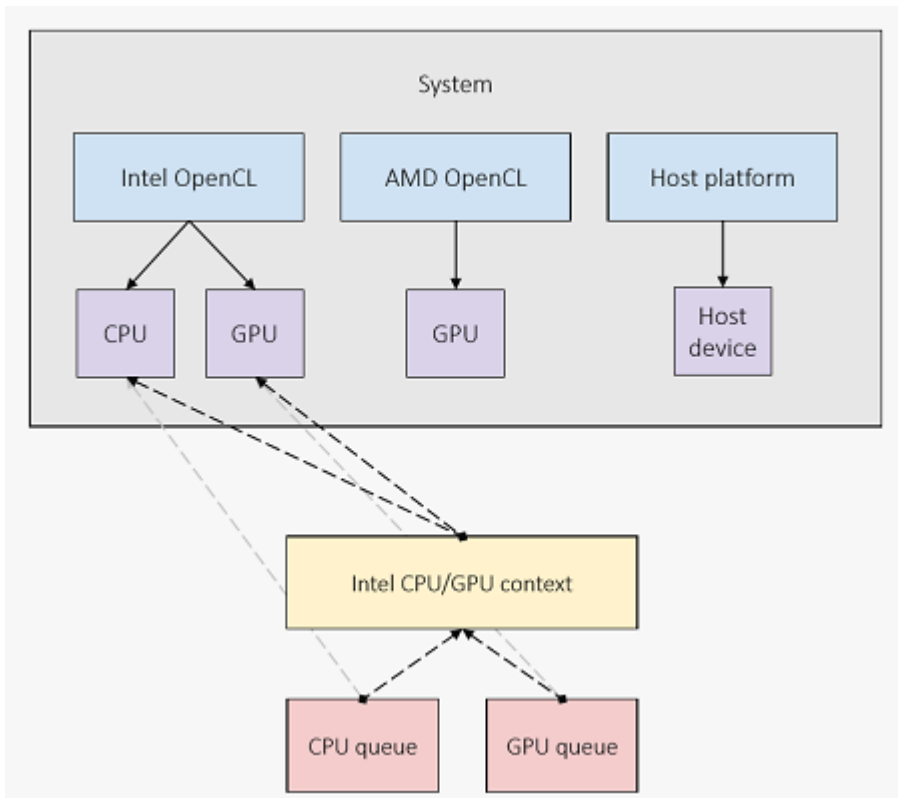
- Every queue requires a context to manage memory allocation and data movement.
- If one is not specified explicitly a queue will create a context implicitly.

# SHARED CONTEXT



- In order to ensure data is efficiently moved between devices in the same platform you can create a common context.

# CREATING QUEUES



- You can then create a queue for each of the devices from the common context.

# CREATING AN IMPLICIT CONTEXT

```
auto defaultQueue = queue{};
```

- A default constructed queue object will use the `default_selector_v` to choose a device and create an implicit context.

# CREATING A CONTEXT FROM DEVICES

```
auto sharedContext = context{{cpuDevice, gpuDevice}};
```

- You can construct a context from a `std::vector` of devices.



## CREATING A CONTEXT FROM A PLATFORM

```
auto sharedContext = context{intelPlatform};
```

- You can construct a context from a platform in which case it will be associated with all of the devices of that platform.

## TARGETING MULTIPLE DEVICES

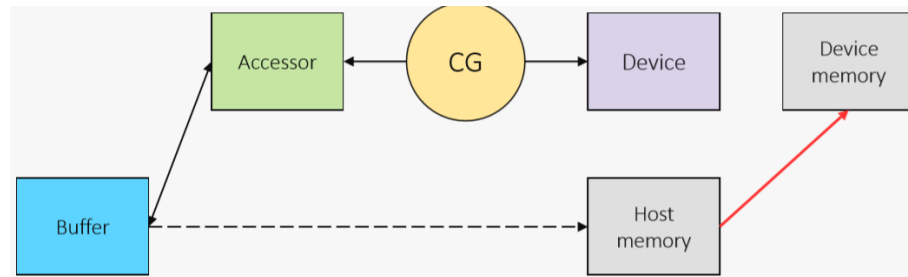
- A single SYCL application will often want to target multiple different devices.
- This can be useful for task level parallelism and load balancing.
- When doing so you want to ensure that data is moved between devices in a context efficiently.

## MOVING BETWEEN DEVICES

- Often in heterogeneous applications it's necessary to move data from one device to another.
- In the USM model this is done explicit via `memcpy` as we've seen before.
- In the buffer/accessor model this is done automatically based on dependency analysis.

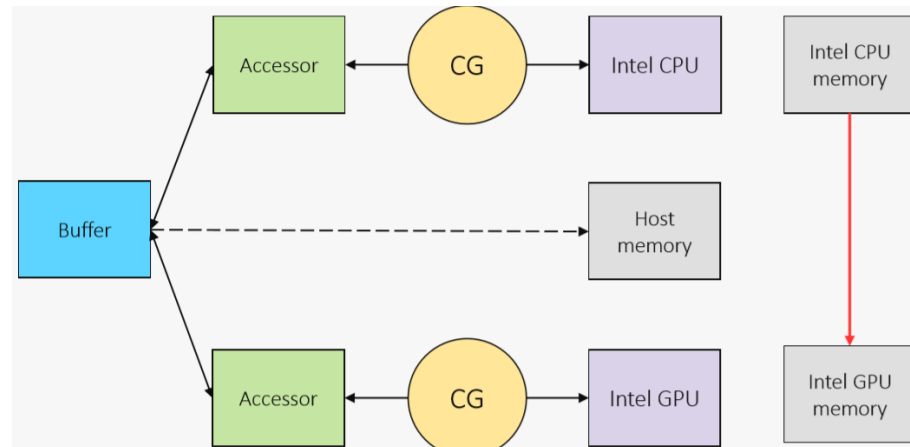
## ACCESSING DATA ON A DEVICE

- Remember that a buffer will move data to a device when required by an accessor.



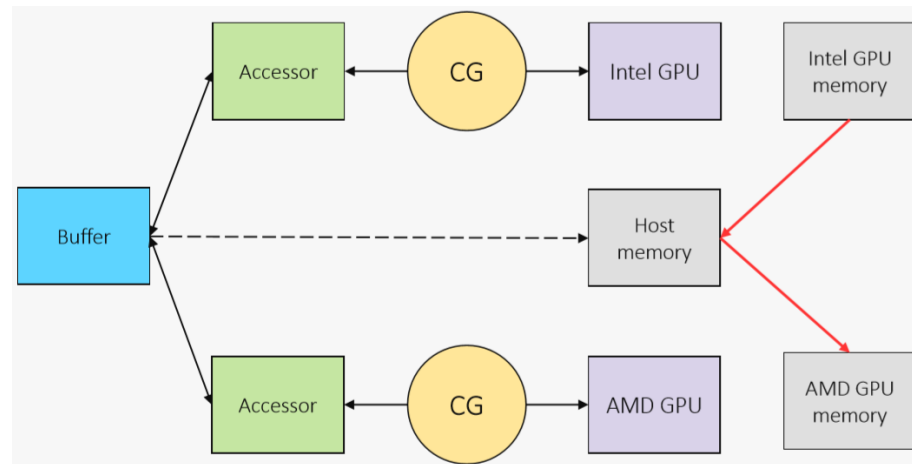
## ACCESSING DATA ON ANOTHER DEVICE (SAME CONTEXT)

- Now if a buffer is accessed on a device when the latest copy of the data is on another device, the data will be moved between the devices.
- If the two devices are of the same context the data can be copied directly.

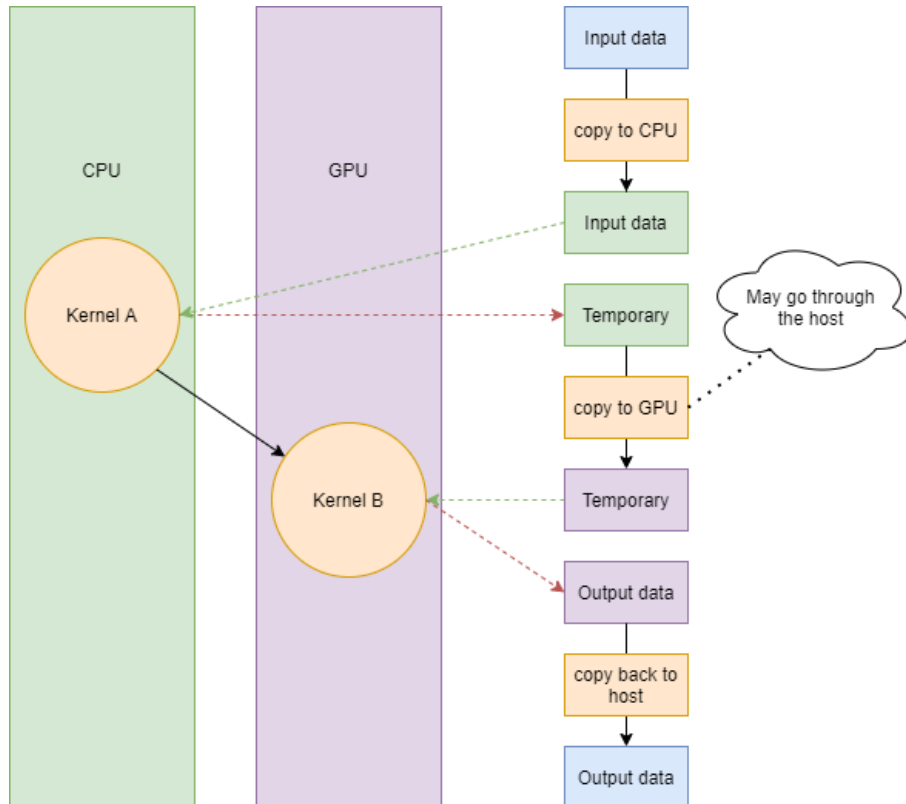


## ACCESSING DATA ON ANOTHER DEVICE (DIFFERENT CONTEXT)

- If the devices are of different contexts the data must be copied via host memory.
- It's important to consider this as it could incur further overhead when moving data between devices.



## MOVING BETWEEN DEVICES (BUFFER/ACCESSOR)



- If a buffer is accessed by kernel functions in two different devices commands are enqueued to automatically move the data to the devices it is being accessed on.
- If both of those devices are associated with the same context (i.e. same vendor) then the copy is direct.
- Otherwise the copy will generally go via the host and has additional overhead.

# QUESTIONS



# EXERCISE

Code\_Exercises/Exercise\_13\_Load\_Balancing/source

Write a SYCL application that splits up a task between two devices.

