Context

- an abstract container that exists on the host
- coordinates the mechanisms for host–device interaction
- manages the memory objects that are available to the devices
- keeps track of the programs and kernels that are created for each device

Command Queues

- Communication with a device occurs by submitting commands to a command queue.
- Once the host decides which devices to work with and a context is created, one command queue needs to be created per device.

Events

- Any operation that enqueues a command into a command queue - that is, any API call that begins with clEnqueue - produces an event.
- Represents dependencies.
- Provide a mechanism for profiling.

Memory Objects

- In order for data to be transferred to a device, it must first be encapsulated as a memory object.
- Two types of memory objects: *buffers* and *images*.
- **Buffers** are equivalent to arrays in C, created using malloc() , where data elements are stored contiguously in memory.
- **Images** are designed as opaque objects, allowing for data padding and other optimizations that may improve performance on devices.
- Memory objects are valid only within a single context.
- Movement to and from specific devices is managed by the OpenCL runtime as necessary to satisfy data dependencies.

Buffer (memory object)

- It is visible for all devices associated with the context.
- Flags specify that the data is read-only, write-only, or read-write.
- Data contained in host memory is transferred to and from an OpenCL buffer using the commands **clEnqueueWriteBuffer()** and **clEnqueueReadBuffer()**, respectively.
- The buffer is linked to a context, not a device, so it is the runtime that determines the precise time the data is moved.

Flush and Finish

- Types of barrier operations for a command queue.
- **clFinish()** function blocks until all of the commands in a command queue have completed (**synchronization barrier**).
- **clFlush()** function blocks until all of the commands in a command queue have been removed from the queue.

OpenCL Program Object

- OpenCL C code (written to run on an OpenCL device) is called a program.
- A collection of functions called kernels, where kernels are units of execution that can be scheduled to run on a device.

1. The OpenCL C source code is stored in a character string. If the source code is stored in a file on a disk, it must be read into memory and stored as a character array.
2. The source code is turned into a program object, **cl_program**, by calling **clCreateProgramWithSource()**.
3. The program object is then compiled, for one or more OpenCL devices, with **clBuildProgram()**. If there are compile errors, they will be reported here.

OpenCL Kernel

- **cl_kernel** object that can be used to execute kernels on a device is to extract the kernel from the **cl_program**.
- After any required memory objects are transferred to the device and the kernel arguments are set, the kernel is ready to be executed.
- Requesting that a device begin executing a kernel is done with a call to **clEnqueueNDRangeKernel()**