IMAGES AND PIPES

OPENCL 2.0 UNIVERSITY TOOLKIT

Zhongliang Chen and Yash Ukidave,
Northeastern University Computer Architecture Research Lab
with
Perhaad Mistry and Dana Schaa, AMD

INSTRUCTOR NOTES

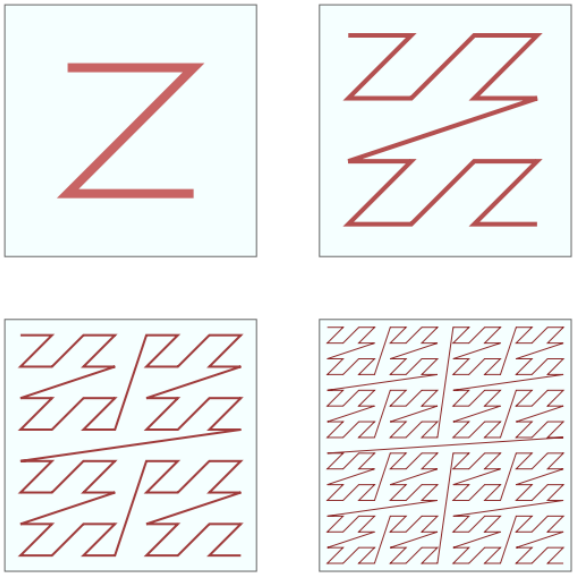
AMD

- We cover the remaining memory objects introduced in the OpenCL 2.0 specification
  - Images
  - Pipes
- These are special purpose memory objects and differ from the C-like buffers
  - Images abstract memory layout to provide simplicity and optimizations
  - Pipes are used to send data between kernel instances in a first-in-first-out manner

IMAGES

- C/C++ arrays and OpenCL buffers (cl_mem) objects provide 1D locality
- Due to graphics workloads, GPUs contain hardware support for:
  - Caching and reading multidimensional data (textures)
  - Drawing interpolated texture vertices
- Hardware support for these features is exposed to programmers via OpenCL images
  - OpenCL images are memory objects optimized for 2D locality
- Adjacent elements not guaranteed to be contiguous in memory
  - Z curve layout of textures in memory provides 2D locality of data

C/C++ 1D locality (row major)layout



Z Curve - 2D locality in layout

Z Curve image from Wikipedia

IMAGES

AMD

- Images are specifically designed to manage graphics data objects
- To allow for hardware abstraction in the physical memory layout, images elements cannot be accessed directly from within the kernel
- Images differ from data buffers in three ways:
  - Opaque data type which cannot be directly viewed using through the device code
  - Multidimensional structures
  - Limited to a range of data types relevant to graphics
- Special hardware provides different operations such as data transforms and filtering
- In OpenCL kernels, the read_image{type} function call is used instead of simply indexing using '[ ]'
  - Usage of read_image and write_image discussed later

BENEFITS OF IMAGES

AMD

- Interpolation
  - Images are accessed using floating point coordinates
    - Either closest pixel can be returned or a linear interpolation
  - Specified in `cl_sampler` object
    - `CL_FILTER_NEAREST` (no interpolation)
    - `CL_FILTER_LINEAR` (linear interpolation)
- Normalized data types
  - Reduces amount of memory used since these data types store floats as a 16- or 8-bit integer in the texture
  - Use floats in a normalized range [0.0-1.0] (unsigned types), [-1.0-1.0] (signed types)

BENEFITS OF IMAGES

AMD

- Out-of-bounds handling
  - Behavior of out-of-bounds accesses are handled in hardware
  - Flags specified when creating cl_sampler
  - Examples
    - CLK_ADDRESS_CLAMP – return 0
    - CLK_ADDRESS_CLAMP_TO_EDGE – return color of pixel closest to out-of-bounds location

BENEFITS OF IMAGES

AMD

- Channels in OpenCL images refer to the primary colors that make up an image
  - Each pixel in a texture can contain 1 to 4 channels (R to RGBA)
  - RGBA: red, green, blue, alpha
  - The color information is stored as float / integer data
- Packing several values (channels) in a pixel this can improve memory bandwidth utilization.
- The number of channels is defined at the creation of the image

CREATING AN IMAGE

- Image declarations consist of descriptors and formats
- Using images in kernels requires the declaration of an image sampler object

```
cl_mem clCreateImage (
        cl_context context,          // OpenCL Context
        cl_mem_flags flags,          // Memory Flags
        const cl_image_format *image_format,   // Image format
        const cl_image_desc *image_desc,       // Image descriptor
        void *host_ptr,              // Host Pointer
        cl_int *errcode_ret)         // Error code
```

IMAGE DESCRIPTORS AND FORMATS

AMD

- Image properties specified in `cl_image_desc` struct, including
  - Image Type – 1D, 2D or 3D Image
  - Image Size – width, height and depth
  - Row and slice pitch
- Image format described in `cl_image_format` struct specifies channel properties and the datatype of each element in the channel
  - Channel properties: The number of channels and the memory layout in which channels are stored in the image
- More detailed syntax can be found in the specification

IMAGE SAMPLERS

AMD

- Image sampler describes how to access an image object
- Samplers specify:
  - The type of coordinate system to access image
  - Options to handle out-of-bounds accesses
  - Interpolation options if an access lies between multiple indices
- A sampler is passed to the kernel just like regular kernel arguments, or can be declared within the kernel
- Creating a sampler on the host uses the call clCreateSampler

```
cl_sampler   clCreateSampler (
   cl_context context,                    // OpenCL context
   cl_bool normalized_coords,             // Use normalized coords?
   cl_addressing_mode addressing_mode,    // Out-of-bounds access behavior
   cl_filter_mode filter_mode,            // Interpolation behavior
   cl_int *errcode_ret)                   // Error code
```

IMAGES IN A KERNEL

AMD

- Images formats are not same as the basic OpenCL types (int, float, char etc.)
- Type qualifiers used for images are `image1d_t`, `image2d_t`, and `image3d_t`

IMAGES IN A KERNEL

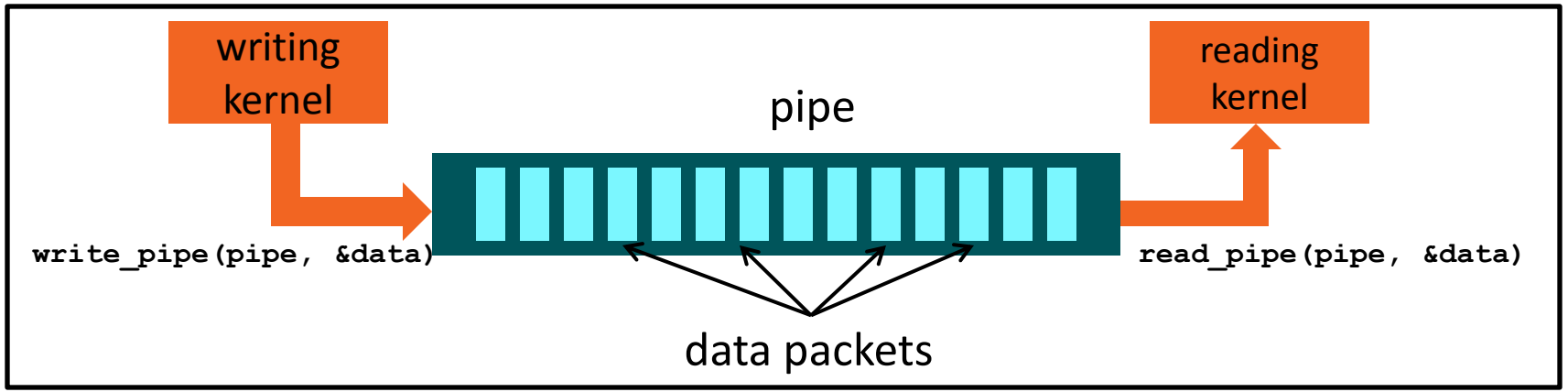
AMD

```
__kernel
void rotation(
    __read_only image2d_t inputImage, __write_only image2d_t outputImage,
    int imageWidth, int imageHeight, float theta)
   {
   // Declaring sampler
   __constant sampler_t sampler =
      CLK_NORMALIZED_COORDS_FALSE | CLK_FILTER_LINEAR | CLK_ADDRESS_CLAMP;

   /* Read the input image */
   float value;
   value = read_imagef(inputImage, sampler, readCoord).x;

   /* Write the output image */
   write_imagef(outputImage, (int2)(x, y), (float4)(value, 0.f, 0.f, 0.f));
}
```

PIPES

AMD

- New memory object introduced in OpenCL 2.0 specification
- Data is organized as *packets* in a FIFO structure
- Pipes have one kernel instance that is a writer and another kernel instance that is a reader
  - The same kernel cannot write and read a pipe
  - Memory consistency is enforced at synchronization points



PIPES

AMD

- Data in pipe is organized as packets
  - Packet can have a type supported by OpenCL C
  - Depth of pipe is defined as the number of packets supported by the pipe
- The pipe can be accessed through the kernel code on the device
- Host creates the pipe object using `clCreatePipe()`
  - Passed as normal kernel argument
  - Host is not allowed to read or write data to the pipe object

```
clCreatePipe(
    cl_context context,                     // Context
    cl_mem_flags flags,                     // Flags same as buffer
    cl_uint pipe_packet_size,               // Packet size
    cl_uint pipe_max_packets,               // Pipe depth
    const cl_pipe_properties *properties, // Pipe properties
    cl_int *errcode_ret)                    // Error code
```

PIPES

- In a kernel, pipes must be declared using the keyword `pipe`, an access qualifier (`read_only` or `write_only`), and the data type of the packets
- An example kernel signature that reads a pipe of type `int` and writes to a pipe of type `float4`:

```
kernel
void foo(read_only pipe int input_pipe,
         write_only pipe float4 output_pipe)
```

PIPES

AMD

- Similar to images, pipes are opaque objects
- Accessing a pipe in a kernel is done using OpenCL C built-in functions
- There are multiple ways to access a pipe; the most basic is using the functions
  - `int write_pipe(pipe p, gentype *ptr)`
  - `int read_pipe(pipe p, gentype *ptr)`
- Both calls take the pipe object and a pointer as parameters
  - Either the data that should be written into the pipe (`write_pipe()`) or the location where the data should be stored after reading (`read_pipe()`)
- Both calls return 0 on success

PIPES – RESERVATION IDENTIFIERS

AMD

- OpenCL C functions exist to reserve space in the pipe ahead of time
  - Accesses are guaranteed to succeed
  - These functions return *reservation identifiers* (`reserve_id_t`) which specify locations within the pipe
- Multiple packets can be reserved to the same reservation identifier using the `num_packets` parameter

```
reserve_id_t
reserve_read_pipe(
   pipe gentype p,
   uint num_packets)
```

```
reserve_id_t
reserve_write_pipe(
   pipe gentype p,
   uint num_packets)
```

PIPES – RESERVATION IDENTIFIERS

AMD

- Reservation identifiers are passed to overloaded versions of `read_pipe()` and `write_pipe()`
  - An index must also be provided specifying the packet location within the reserved space
- When reservation identifiers are used, an additional blocking call (`commit_read_pipe()` or `commit_write_pipe()`) is required to ensure that the operation has successfully completed

```
int
read_pipe(
   pipe gentype p,
   reserve_id_t reserve_id,
   uint index,
   gentype *ptr)
```

```
void
commit_read_pipe(
   pipe gentype p,
   reserve_id_t reserve_id)
```

PIPES

AMD

- Additional versions of pipe reservation and commit calls exist on a work-group granularity
  - `work_group_reserve_read_pipe()` and `work_group_reserve_read_pipe()`
  - `work_group_commit_read_pipe()` and `work_group_commit_read_pipe()`
- OpenCL C calls also exist to determine the number of packets in the pipe (`get_pipe_num_packet()`) and the size of the pipe (`get_pipe_max_packets()`)

SUMMARY

AMD

- An image is a data type that enables optimizations by hiding the memory layout from the programmer
  - Interpolation
  - Normalization
  - Bounds-handling
- Pipes make it easier to pass data between kernel instances for certain classes of algorithms
- Both images and pipes are opaque types, accessible only by intrinsic OpenCL C functions