

OPENCL

Episode 3 - Building an OpenCL Project

David W. Gohara, Ph.D.

Center for Computational Biology

Washington University School of Medicine, St. Louis

email: sdg0919@gmail.com twitter: iGotchi

THANK YOU



Q & A

- Double-precision arithmetic
- Object Oriented programming
- Global vs local work group sizes
- Classes of scientific problems addressable with OpenCL

Q & A - DOUBLE PRECISION

- Optional in OpenCL
 - `#pragma OPENCL EXTENSION cl_khr_fp64 : enable`
- Support is implementation and card dependent
 - If implemented it must conform to the IEEE-754 standard
- Performance is reduced relative to single-precision

Q & A - OO PROGRAMS

- OpenCL can be called from almost any programming language
 - Catch: The language must be able to interface with C stubs
- OpenCL does not support passing of “objects” into kernels
 - Structures may be passed in
 - Data layout is extremely important

Q & A - WORK GROUP SIZES

- On the CPU the local group size is always one
 - Synchronization points are no-ops and are too expensive to implement
- Determining local work group size can be trial and error
- Typically the local work group size should be no smaller than the size of a warp (or wavefront)
- Powers of two when possible!
- Max local work group size ≤ 512
- Dimensioning is really only important for simplifying indexing

Q & A - PROBLEM CLASSES

- FFTs
- BLAS/LAPACK type operations
- Monte Carlo
- PDEs

Q & A - PROBLEM CLASSES

- Not all algorithms (or implementations) are optimal on a GPU
- Problems often need to be re-factored or data structures modified
- Computations DO NOT need to run in a single kernel or queue call

Q & A - PROBLEM CLASSES

- Each portion of the calculation can be run as a separate kernel
- Each kernel is queued in order, possibly within a loop
- Checks for early exit require data transfers

```

$$\mathbf{r}_0 := \mathbf{b} - \mathbf{A}\mathbf{x}_0$$

$$\mathbf{p}_0 := \mathbf{r}_0$$

$$k := 0$$
repeat  

$$\alpha_k := \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T \mathbf{A} \mathbf{p}_k}$$

$$\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k \mathbf{p}_k$$

$$\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k \mathbf{A} \mathbf{p}_k$$
if  $\mathbf{r}_{k+1}$  is sufficiently small then exit loop end if  

$$\beta_k := \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}$$

$$\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k \mathbf{p}_k$$

$$k := k + 1$$
end repeat  
The result is  $\mathbf{x}_{k+1}$ 
```

<http://bit.ly/QDB1a>

OPENCL EXAMPLE

- Mac OS X Snow Leopard
- Xcode tools installed
- Simple example to get you familiar with the actual code and tool chain

DISCOVERING DEVICES

```
clGetDeviceIDs(NULL, CL_DEVICE_TYPE_CPU, 1, &device, NULL);
```

CL_DEVICE_TYPE_GPU
CL_DEVICE_TYPE_ACCELERATOR
CL_DEVICE_TYPE_DEFAULT
CL_DEVICE_TYPE_ALL

of devices

returned device(s)

DEVICES PROPERTIES

```
clGetDeviceInfo(device, CL_DEVICE_VENDOR, sizeof(vendor_name),  
                vendor_name, &returned_size);
```



CL_DEVICE_MAX_MEM_ALLOC_SIZE
CL_DEVICE_GLOBAL_MEM_SIZE
CL_DEVICE_QUEUE_PROPERTIES
CL_DEVICE_MAX_WORK_ITEM_SIZES
CL_DEVICE_EXTENSIONS
...


BUILDING PROGRAMS

```
clBuildProgram(program[0], 0, NULL, NULL, NULL, NULL);
```

```
char buffer[2048];
```

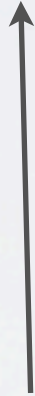
```
clGetProgramBuildInfo(program, device_id, CL_PROGRAM_BUILD_LOG,  
    sizeof(buffer), buffer, &len);
```

CL_PROGRAM_BUILD_STATUS
CL_PROGRAM_BUILD_OPTIONS



MEMORY BUFFERS

```
a_mem = clCreateBuffer(context, CL_MEM_READ_ONLY, buffer_size,  
                           NULL, NULL);
```



CL_MEM_READ_WRITE
CL_MEM_WRITE_ONLY
CL_MEM_USE_HOST_PTR
CL_MEM_ALLOC_HOST_PTR
CL_MEM_COPY_HOST_PTR

MEMORY BUFFERS

```
clEnqueueWriteBuffer(cmd_queue, a_mem, CL_TRUE, 0, buffer_size,  
                    (void*)a, 0, NULL, NULL);
```

CL_FALSE



EXECUTION/READ

```
clSetKernelArg(kernel[0], 0, sizeof(cl_mem), &a_mem);
```

```
clEnqueueNDRangeKernel(cmd_queue, kernel[0], 1, NULL,  
                        &global_work_size, NULL, 0, NULL, NULL);
```

```
clEnqueueReadBuffer(cmd_queue, ans_mem, CL_TRUE, 0, buffer_size,  
                    results, 0, NULL, NULL);
```

XCODE DEMO

MORE INFORMATION

- MacResearch.org
 - OpenCL - <http://www.macresearch.org/openccl>
 - Amazon Store - <http://astore.amazon.com/macreseorg-20>
- Sparse Matrix-Vector Multiplication - <http://bit.ly/13rOnM>
- Mixed Precision Arithmetic - <http://bit.ly/4c0eAc>