Heterogeneous Parallel Programming using OpenCL

Mandar Gurav
Indian Institute of Technology Bombay
Mumbai

Code Structure

- Get a list of available platforms
- Select device
- Create Context
- Create command queue
- Create memory objects
- Read kernel file
- Create program object
- Compile kernel
- Create kernel object
- Set kernel arguments
- Execute kernel
- Read memory object
- Free objects

Get a list of available platforms

cl_int **clGetPlatformIDs**(cl_uint num_entries,

    cl_platform_id *platforms,

    cl_uint *num_platforms)

Select device

cl_int **clGetDeviceIDs**(cl_platform_id platform,

cl_device_type device_type,

cl_uint num_entries,

cl_device_id *devices,

cl_uint *num_devices)

Create Context

```
cl_context clCreateContext( cl_context_properties *properties,
    cl_uint num_devices,
    const cl_device_id *devices,
    void *pfn_notify (
                const char *errinfo,
                const void *private_info,
                size_t cb,
                void *user_data
                ),
    void *user_data,
    cl_int *errcode_ret)
```

Create command queue

cl_command_queue **clCreateCommandQueue**(cl_context context,

    cl_device_id device,

    cl_command_queue_properties properties,

    cl_int *errcode_ret)

Create memory objects

cl_mem **clCreateBuffer** (cl_context context,

    cl_mem_flags flags,

    size_t size,

    void *host_ptr,

    cl_int *errcode_ret)

Write to memory object

cl_int **clEnqueueWriteBuffer** (cl_command_queue command_queue,

cl_mem buffer,

cl_bool blocking_write,

size_t offset,

size_t cb,

const void *ptr,

cl_uint num_events_in_wait_list,

const cl_event *event_wait_list,

cl_event *event)

Create Kernel program: online

cl_program **clCreateProgramWithSource** (cl_context context,

    cl_uint count,

    const char **strings,

    const size_t *lengths,

    cl_int *errcode_ret)

Create Kernel program: offline

cl_program **clCreateProgramWithBinary** (cl_context context,

    cl_uint num_devices,

    const cl_device_id *device_list,

    const size_t *lengths,

    const unsigned char **binaries,

    cl_int *binary_status,

    cl_int *errcode_ret)

* can store binaries using **clGetProgramInfo** with CL_PROGRAM_BINARIES flag

Build Kernel Program

```
cl_int clBuildProgram ( cl_program program,
   cl_uint num_devices,
   const cl_device_id *device_list,
   const char *options,
   void (*pfn_notify)(cl_program, void *user_data),
   void *user_data)
```

Create OpenCL Kernel

```
cl_kernel clCreateKernel ( cl_program  program,
    const char *kernel_name,
    cl_int *errcode_ret)
```

Set OpenCL kernel argument

cl_int **clSetKernelArg** (cl_kernel kernel,

cl_uint arg_index,

size_t arg_size,

const void *arg_value)

Launch Kernel: single thread

cl_int **clEnqueueTask** (cl_command_queue command_queue,

cl_kernel kernel,

cl_uint num_events_in_wait_list,

const cl_event *event_wait_list,

cl_event *event)

Launch Kernel: multiple threads

cl_int **clEnqueueNDRangeKernel** (cl_command_queue command_queue,

    cl_kernel kernel,

    cl_uint work_dim,

    const size_t *global_work_offset,

    const size_t *global_work_size,

    const size_t *local_work_size,

    cl_uint num_events_in_wait_list,

    const cl_event *event_wait_list,

    cl_event *event)

Read from memory object

cl_int **clEnqueueReadBuffer** (cl_command_queue command_queue,

cl_mem buffer,

cl_bool blocking_read,

size_t offset,

size_t cb,

void *ptr,

cl_uint num_events_in_wait_list,

const cl_event *event_wait_list,

cl_event *event)

Free objects

```
ret = clFlush(command_queue);
ret = clFinish(command_queue);
ret = clReleaseKernel(kernel);
ret = clReleaseProgram(program);
ret = clReleaseMemObject(memobj);
ret = clReleaseCommandQueue(command_queue);
ret = clReleaseContext(context);
```

Thank you.

mandar.hpsc@gmail.com