Видео-карточки. Теория

Александр Сергеев

1 Общие штуки

Платформы(Intel, AMD, NVIDIA) содержат девайсы(видяха 1, видяха 2)

```
clGotPlatformIDs(NULL, 0, &sz); //return memory size needed
   in sz
clGotPlatformIDs(buffer, buffer_size, NULL); //return
   platform list in buffer
```

Возвращает переменное количество аргументов

```
#include <CL/cl.h>  //minimal needed header
```

Виды функций в cl:

1. возвращает код ошибки
2. функции *clCreate**: возвращает объект, код ошибки по указателю

```
clGetPlatformInfo(...); //get platform info
clGetDeviceIDs(platform, ...);
clGetDeviceInfo(device, ...);
```

2 Создание контекста

```
clCreateContext(...);
id = clCreateProgramWithSource(...);    //load files in
   context
err = clBuildProgram(id, device_list, build_options, ...);
   //compile file $id for devices from $device_list and link
   it to program $id, build_options = "...", not NULL
clGetProgramBuildInfo(...);
```

3 Код

```
kernel void add(global const int *a, global const int *b,
    global int *c) {
     *c = *a + *b;
}
```

kernel – точка входа

```
clCreateKernel(...);
```

4 Память

size_t на девайсе != size_t на хосте

```
cl_mem buf = clCreateBuffer(...);
clCreateCommandQueue(device, flags); //flags: profiling_info
    -- enable stats, out_of_order_execution_enable -- do not
    use it
clSetKernelArg(id, arg_n, buf, buf_size);
clEnqueueWriteBuffer(buf, data, data_size, ...);  //flag:
    blocking_write. If not,
clEnqueueNDRangeKernel(dimentions, &global_work_size, offset=
    NULL, local_work_size=NULL);             //dimentions = 1,
     global_work_size = 1,
clEnqueueReadBuffer(...);  //flag: blocking_read
clReleaseMemObject(buf);
```

Можно сделать запись и исполнение неблокирующими, а чтение – блокирующим
Блокирующие операции запускают очередь
Т.к. действия выполняются последовательно, то мы заблокируемся до конца исполнения
CreateBuffer – ленивый, т.е. память создается в момент использования