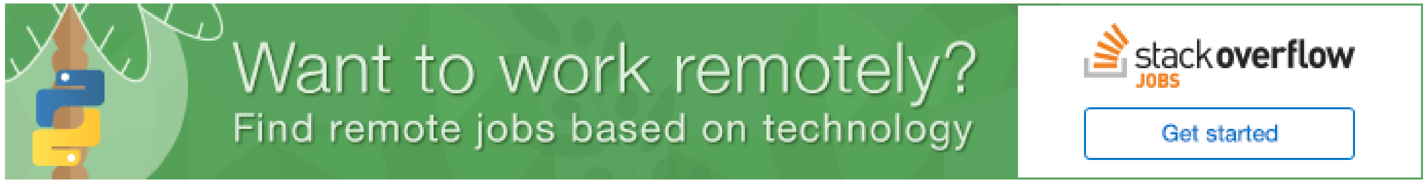
Sharing OpenCL Kernel Data



I have 2 OpenCL kernels, `run_kernel` and `apply_kernel` that I want completed sequentially one after the other, a few times. The output of `run_kernel` contains some of the input for `apply_kernel`, but I'm not sure how to implement this.

Currently, I have a single `cl_mem` buffer named `d_vertexBuffer` that I filled with the data I want to give `run_kernel`, and it does its thing correctly. I set the kernel arg like this:

```
error = clSetKernelArg(run_kernel, 0, sizeof(cl_mem), (void*) &d_vertexBuffer);
```

I tried setting `apply_kernel` to use the same `d_vertexBuffer`, but I'm guessing this messes up `run_kernel` accessing to it, since the OpenCL code is getting NaN whenever it tries to access the buffer. I set the `apply_kernel` like this:

```
error = clSetKernelArg(apply_kernel, 0, sizeof(cl_mem), (void*) &d_vertexBuffer);
```

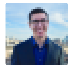
I create the `d_vertexBuffer` like this:

```
d_vertexBuffer = clCreateBuffer(context, CL_MEM_READ_WRITE | CL_MEM_COPY_HOST_PTR,
vertexBufferSize, h_vertexBuffer, &error);
```

In order to run these kernels multiple times, I have a `for` loop that enqueues the kernel in my command queue. Obviously this must not be the correct way to do it. How would I make it so that the two kernels are able share data?

c opencl

asked Jul 29 '13 at 15:40


K. Barresi
690 1 15 36

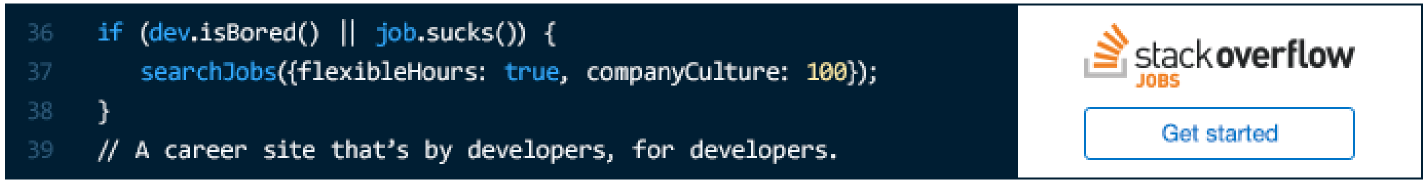
2 Answers

By the sounds of it, you want the ability to append the important output from `run_kernel` onto the end of `d_vertexBuffer`. You could make `d_vertexBuffer` large enough to store the normal input values (`vertexBufferSize`) plus the extra vertices from the output of `run_kernel`. `run_kernel` copies the part of its output that matters for `apply_kernel` into the section of `d_vertexBuffer` above `vertexBufferSize`

answered Jul 29 '13 at 16:41

chippies
1,348 5 15

The problem is that the `d_vertexBuffer` contains `float4` elements, and I need to edit each elements for every iteration of the two kernels. If I were to just tack on new elements for every iteration, it would become unreasonably large. – K. Barresi Jul 29 '13 at 16:45



```
if (dev.isBored() || job.sucks()) {
    searchJobs({flexibleHours: true, companyCulture: 100});
}
// A career site that's by developers, for developers.
```

The problem ended up being unrelated; I was accidentally using a 2-index global work size in the `apply_kernel` when I only wanted 1, so it was throwing out NaN,

answered Jul 29 '13 at 19:34

K. Barresi
690 1 15 36