OpenCL event profiling

Jump to bottom

zoran edited this page on Sep 9 · 1 revision

For newcomers, the PyOpenCL documentation won't immediately come together. The fact that the docs are somewhat sparse unfortunately complicates matters. this post is derived from a Piazza answer to the question: "How do I time the execution of OpenCL events when using PyOpenCL?" What follows is the answer, almost verbatim.

Your goal is to time the execution of an OpenCL kernel. Let's start with an overview of how to do that. First, what are the key execution details? Here's a list, followed by the answers to them:

- How is the kernel invoked after being built?
- 'Where' does the kernel function run?

The kernel code is defined by creating an instance of the `cl.Program` class. By itself, this does nothing. `cl.Program` classes have a method called `build()`, which builds the binary(s) for the kernel code defined in the class instance.

As you've seen in the demo notebook, you can call the kernel function by using the kernel name. So if your kernel code function is named `doublify`, then you could call it simply by running `prg.doublify`, where `prg` is the instance name. Here's a link to the docs for `cl.Program`.

So far, the kernel has been built, and we have a means by which to call the function. The next question is, 'where' does it run? In the demo notebook, the function `doublify` is called like so:

```
prg.doublify(queue, a_np.shape, None, a_g, res_g)
```

When you call a function in this manner, an OpenCL event is created within the current running context, which has an associated command queue. Calling the function this way doesn't make that immediately obvious. PyOpenCL allows you to identify and name kernel execution events using the `cl.Event` class. To do this, just name your function call:

```
evt = prg.doublify(queue, a_np.shape, None, a_g, res_g)
evt.wait()
```

You'll notice that in the docs for `cl.Event`, one of the methods available is `get_profiling_info()`. You'll notice that the method description has a note:

"See `profiling_info` for values of param. See `profile` for an easier way of obtaining the same information."

Here is where the documentation sort of breaks down and I'm guessing is where you got stuck. If you navigate to the link for `profiling_info`, you're presented with an extremely obscure and rather empty looking description of that class. You'll see there are several class attributes, two of which are relevant to timing: `START` and `END`.

To time execution of an OpenCL event, you have to make use of these two. Since there are no examples nor a clear description of how to use them, here's how to time the event:

```
evt = prg.doublify(queue, a_np.shape, None, a_g, res_g)
evt.wait()
exec_time = evt.profile.end - evt.profile.end
```

The time will be returned in nanoseconds.

Footnote: The call `evt.profile.end` isn't completely out of the blue. You might have noticed that there are two classes defined under the *Events* heading in the docs. The second, smaller class is `cl.ProfilingInfoGetter`, whose description actually tells you exactly what to do with those `START` and `END` attributes of the `profiling_info` class. Here's what it says:

"Lower case versions of the profiling_info constants may be used as attributes on the attribute profile of this class to directly query `profiling info`." For example, you may use evt.profile.end instead of evt.get_profiling_info(pyopencl.profiling_info.END).

As I noted in the first recitation, the docs are hard to parse, but once you get used to navigating them, you'll find answers to every question, although it might take some digging. I hope this answer also gives everyone an insight in how to interpret the docs, which is a very useful habit to form.

EECS E4750: Heterogeneous Computing for Signal and Data Processing (Fall 2021)

Pages 18

E4750 Course Wiki Home

1. Home Page
2. Tutorials

- Google Cloud
  - Google Cloud VM Setup
  - GUI Installation
- Code
  - Python
  - PyCUDA Tutorial
  - Indexing in CUDA & OpenCL

3. Concepts and Additional How-Tos

- CUDA Profiling
- CUDA Cores v. Threads
- Data Types
- Timing execution in PyOpenCL

4. Assignments (distributed from the Code section)
   1. Assignment 1
   2. Assignment 2

Clone this wiki locally

```
https://github.com/eecse4750/e4750_2021Fall_students_repo.wiki.git
```