


[x Dismiss](#)


Join the Stack Overflow Community

Stack Overflow is a community of 7.3 million programmers, just like you, helping each other.
Join them; it only takes a minute:

[Sign up](#)

Write multiple kernels or a Single kernel


[Get started](#)



Want to work remotely?
Find remote jobs based on technology

Suppose that I've two big functions. Is it better to write them in a separate kernels and call them sequentially, or is better to write only one kernel? (I don't want to read the data back and force form between host and device in between). What about the speed up if I want to call the kernel many times?

opengl

edited Mar 1 '12 at 10:06



[talonmies](#)

54.2k 17 109 164

asked Feb 29 '12 at 18:46



[Damoon](#)

144 2 13

have a look here, similar discussion: stackoverflow.com/questions/9208535/how-to-handle-a-variable-number-of-algorithms-in-a-kernel – [rdoubleui](#) Mar 1 '12 at 8:49

3 Answers

One thing to consider is the effect of register pressure on hardware utilization and performance.

As a general rule, big kernels have big register footprints. Typical OpenCL devices (ie. GPUs) have very finite register file sizes and large kernels can result in lower concurrency (fewer concurrent warps/wavefronts), less opportunities for latency hiding, and poorer overall performance. On the other hand, kernel launch overheads are pretty low on most platforms, so if your algorithm doesn't have an enormous amount of state to save between "phases" of execution, the penalty of using multiple kernels can be rather low.

Using multiple kernels also has another side benefit -- you get implicit synchronization between all work units for free. Often that can eliminate the need for atomic memory operations and synchronization primitives which can have a negative impact on code performance.

The ultimate guide should be measured performance. There is no universal rule-of-thumb for this sort of things. Benchmarking is the only way to know for sure.

answered Mar 1 '12 at 10:16



[talonmies](#)

54.2k

17

109

164

Apple agree: [developer.apple.com/library/mac/documentation/Performance/...](https://developer.apple.com/library/mac/documentation/Performance/) – [Ben-Uri](#) Jul 6 '14 at 13:03



Grow your app revenue with house ads

Learn more



AdMob by Google

In general this is a question of (maybe) slightly better performance vs. readability of your code. Copying buffers is no issue as long as you keep them within the same context. E.g. you could set one output buffer of a kernel to be an input buffer of the next kernel, which would not involve any copying.

answered Mar 1 '12 at 8:54



[rdoubleui](#)

2,240 4 20 47

The proper way to code in OpenCL is to separate your code into parallel tasks, and each of them is a kernel. This is, each "for loop" should be a kernel. Some times one single CPU code function could result in a 4 kernel implementation in OCL.

If you need to store data between kernel executions just use OpenCL buffers and do not copy to host (this solves the DEVICE<->HOST bottleneck).

If both functions act to different data you could propably write a single kernel, but that depends on the complexity of the operation being run.

answered Mar 1 '12 at 15:24



[DarkZeros](#)

6,357 1 13 24