Google blocked OpenCL on Nexus with Android 4.3

Important: this is only for Google-branded Nexus phones - other brands are free to do what they want, and they most-probably will.

Also important: this doesn't mean that OpenCL on Android devices will be over, but that there is a bump in the road now Google tries to lock-in customers to their own APIs.

The big idea behind OpenCL is that higher level languages and libraries can be built on top of it. This is exactly what was done under Android: RenderScript Compute (a higher-level language) was implemented using OpenCL for ARM Mali GPUs.

Having OpenCL drivers on Android has several advantages, such that OpenCL can directly be used on Android and that there is room for other high-level languages that have OpenCL as back-end. Especially the latter is what probably made Google decide to cripple the OpenCL-drivers.

Google seems to be afraid of competition, and that's a shame, as competition is the key factor that drives innovation. The OpenCL community is not the only one complaining about Google's intentions concerning Android. Read page 3 of that article to understand how Google is controlling handset-vendors and chip-makers.

Google's statement

In February OpenCL drivers were discovered on two Nexus tablets using a MALI T604 GPU. Around the same time there was one public answer from Google employee Tim Murray (twitter) why Google did not want to choose OpenCL:

1) The answer is that Android's needs are very different than what OpenCL tries to provide.

2) OpenCL uses the execution model first introduced in CUDA. In this model, a kernel is made up of one or many groups of workers, and each group has fast shared memory and synchronization primitives within that group. What this does is cause the description of an algorithm to be intermingled with how that algorithm should be scheduled on a particular architecture (because you're deciding the size of a group and when to synchronize within that group).

3) That's great when you're writing for one architecture and you want absolute peak performance, but it gets peak performance at the expense of performance portability. Maybe on your architecture, you have enough registers and shared memory to run 256 workers per group for best performance, but on another architecture, you'd end up with massive register spills with anything above 128

workers per group, causing an 80% performance regression. Meanwhile, because your code is written explicitly for 256 workers per group, the runtime can't do anything to try to improve the situation on another architecture--it has to obey what you've written. This sort of situation is common when moving from architecture to architecture on the desktop/HPC side of GPU compute.

4) On mobile, Android needs performance portability between many different GPU and CPU vendors with very different architectures. If Android were to rely on a CUDA-style execution model, it would be almost impossible to write a single kernel and have it run acceptably on a range of devices.

5) RenderScript abstracts control over scheduling away from the developer at the cost of some peak performance; however, we're constantly closing the gap in terms of what's possible with RenderScript. For example, ScriptGroup, an API introduced in Android 4.2, is a big part of our plans to further improve GPU code generation. There are plenty of new improvements coming that make writing fast code even easier, too.

This is no good explanation, but just tries to convince people who have little or no experience with OpenCL or CUDA.

1) Google targets on a subset of algorithms that OpenCL can tackle, then says it only needs the subset. This is where the big disagreement is, and now it is hard, or even impossible, to implement certain algorithms in an optimal way. Google did not foresee a need for FFT, but you can find an implementation anyway.

2) This is exactly the big difference between OpenCL and RenderScript. The RenderScript host-driver tries to take care of the number of GPU/CPU cores being used by the software, but in return you cannot specify the worker-configuration.

3) In OpenCL it is also possible to leave the workgroup-configuration to the driver, just like in RenderScript. The big advantage is that it scales very well and gets better with improved drivers. The bad thing is that not all algorithms can be implemented, or at least not efficiently. The examples Tim gives here, would not get through our code-review!

4) True, this is the continuous discussion on advancing OpenCL. As the processors advance each year and we cannot look into the future, it is quite something to say that there is a solution already.

5) But it still is not possible to get the ID of the worker in Renderscript. OpenCL architects are working on improving the solution, but if algorithms could be implemented without knowing the worker-ID then of course this would never been introduced to the OpenCL model. Tim is claiming that OpenCL/CUDA's model does not work, but does not provide a solution.

See this Linkedin-thread for further discussion.

Also read the reply from Scott Le Grand in the comments below.

Google blocked OpenCL in Android 4.3

It was announced on Twitter that on Android 4.3 OpenCL stopped working. An ARM intern told that Google blocked access to the OpenCL-driver, such that it can now only be used by the RenderScript-compiler. The block was quite harsh, as it blocked at the LLVM-frontend level in a way that an easy hack was not possible. ARM could not do much to reverse it, as it is the choice of Google and the block is implemented in Android's core-libraries.

So Google blocked OpenCL drivers on purpose, overruling the decision of their hardware-partners to provide OpenCL-support for their customers.

The only way out seems to go to Linux. An OpenCL-driver for Ubuntu-on-ARM on the Chromebook will be released via the ARM Mali developer webpage in the coming month, probably at the SDK-page.

The exact state is in the below table.

Why is Google against OpenCL?

We can speculate a little around the reasons behind not opting for OpenCL:

- OpenCL is an open standard. Google cannot have the influence it wants. Its future depends on the opinion of many, not on theirs alone.
- On top of OpenCL various high-level languages can be built, also the high-level languages from competitors.
- OpenCL is initiated by Apple. Apple is a competitor of Google in the smartphone OS market. Even though I think it shows strength if a company supports open standards, they probably don't think that themselves.
- Android is fragmented. Google tries to fix this image of the fragmented android space, and must show teeth&balls. OpenCL is the victim.
- Avoiding (accelerated) software to be ported to other operating systems, makes Android unique by its apps.
- Pushing a proprietary standard is hard, even though getting support for open standards is harder to accomplish. Once a company gets their own proprietary software accepted, the profits raise. In other words: vendor-lockin sells.

So, it actually could be that it is not specifically OpenCL, but any open standard that can be blocked without to much trouble. They tried replacing OpenGL with RenderScript (the RenderScript-API was much larger before), but they did not succeed.

Why not simply choose RenderScript?

RenderScript has various design-decisions which various experienced developers do not agree upon. If there is competition, the bad decisions will be reversed sooner. One unpopular design decision is that RenderScript does not inform the software if it is running on a CPU or GPU, and therefore cannot be optimized for speed. Compare it to driving a car and not knowing you are at an urban road or at a 16-lane motorway - do you drive the same in both situations or optimize your driving-style?

Here is a list of blogs describing RenderScript from an OpenCL perspective:

[list1]

CodeDivine: Renderscript from the perspective of an OpenCL/CUDA/C++ AMP programmer]
CodeDivine: More thoughts on Renderscript - based on this discussion.]
Scarpino: Ten Reasons Why Android Should Support OpenCL]
Bad Logic Games: RenderScript.]

[/list1]

If you want to learn more about RenderScript Compute The official documents page is the place to go since it's not possible to find much on the net. In any case, please try it out and let me know in the comments in what cases you would prefer RenderScript over OpenCL, and when would you prefer OpenCL over RenderScript.

You can also let me know in the comments what you think my readers should also read. I'll add the info here.

Share your opinion

Once a new bug-report gets submitted for the OpenCL-driver being blocked on Android 4.3, I'll add it here.

If you want to add to the discussion on OpenCL-on-Android, please add your comments here:

[list1]

- First bug-report at Android-code for getting OpenCL.]
- LinkedIn discussion at OpenCL Developers group]

Discussion in Spanish by folks who translated this post.]
Stackoverflow-question. Has been closed, but still active in upvoting/downvoting and putting comments.]
New Stackoverflow-discussion.]

- In the comments below.

[/list1]

To get OpenCL back on Android, we need to complain, cry and whine. This is how OpenGL got back, and unfortunately the only way there is. If we succeed then innovative projects like AccelerEyes Mobile, ViennaCL and VexCL may end op on Android. If not, then the project developers need to rewrite all the code, but only if the developers want to create two different source-trees and if the limitations of RenderScript do not obstruct porting the already efficient code.