First, I investigated whether the OpenGL shared libraries from the MotoX phone would solve any of the forceful reboot issues I was experiencing on the Nexus 4. I transferred over the shared libraries (they had the same shared libraries for OpenGL, but they were not binary equivalents), and discovered that the Nexus 4 booted up as normal. However, on attempting to run a long running OpenCL kernel, I discovered that the forceful reboot problem still existed, taking (as before) roughly 10 seconds before the device would reboot.

I then investigated more information about the GPU hardware information for the MotoX. I had trouble finding any information directly from Qualcomm, but I was able to find information on the chipset used by the MotoX (the MSM8960DT) from a third party device specification database (PDAdb.net). In particular, the chipset has a 400 MHz quad-core Adreno 320 MP4 GPU. I wasn't able to find information about the GPU device memory [1].

I then started modifying the OpenCL code for running the AES encryption algorithm. In particular, previously it used a global work size of (N / 16) where N is the size of the input array. However, this could result in extra overhead in thread scheduling, depending on how the OpenCL runtime decides to execute the OpenCL kernels. I modified the algorithm such that the input array is split up evenly amongst the global work-size. In particular, all OpenCL instances get some multiple of 16 bytes on which to operate, all of which are equally sized except one OpenCL instance which will be allocated whatever remains of the input array (i.e. given N = 128 bytes, G = 3, then 2 instances will operate on floor((N/16)/G)*G*16 = 48 bytes and 1 instances will operate on N - floor((N/16)/G)*G*16 = 32 bytes). I performed timing experiments on a 128MB input array, and varied the global work size from 1 up to 16.

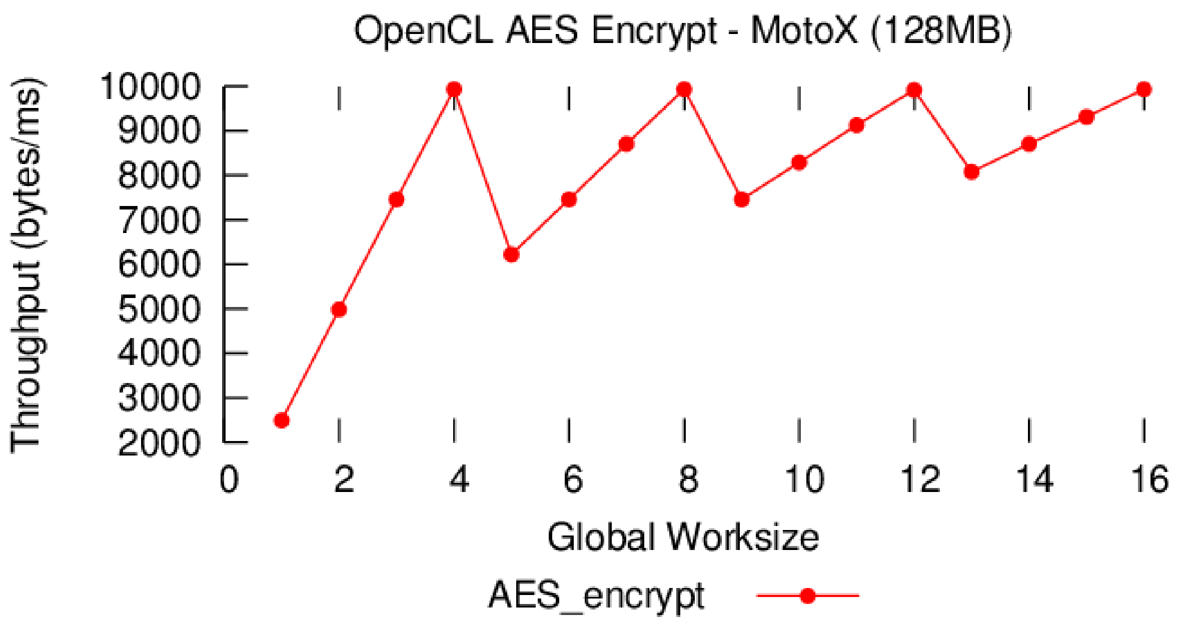

Figure 1: AES encrypt throughput over variable global work-size

From the graph, **we can see that the maximum throughput achieved is 9929.65 bytes/ms (~ 9.47 MB/s) when using a global work-size of 4.**

Going from a global work-size of one up to four, we see a linear speedup in encryption throughput (going from global work-size of 1 to 2 we see a 2487.03 bytes/ms speedup, 2 to 3 we see 2475.66 bytes/ms, and 3 to 4 we see 2472.17 bytes/ms). These linear speedups going from a global work-size of 1 up to 4 are consistent with the Adreno 320 having 4 cores (utilizing an additional core for each increase in global work-size). When we reach a global work-size of five, we see the worst decrease in encryption throughput (3712.83 bytes/ms). This is most likely due to the OpenCL runtime scheduling 4 simultaneous OpenCL kernel instances that operate on 1/5 of the input array, with a single OpenCL instance running on the remaining 1/5 only after the first 4 complete.

The throughput degrades the most for global work-sizes that that are 1 modulo 4 since given that instances will tend complete in roughly the same time (same data size and instructions executed), there will tend to a point at which only one GPU core will be utilized, thus reducing parallelism (similarly for 2 modulo 4, and 3 modulo 4). As we increase in global work-size, the degradation is less since the point at which we aren't maximally using all 4 GPU cores operates over less of the input array (in particular, N/G).

As suggested by Sahil, I investigated the OpenCL code to ensure operations were not performed repeatedly on global memory, since this is the slowest memory in the OpenCL memory hierarchy. Indeed, the OpenCL code uses local variables for repeatedly mutating memory before writing to the global memory buffer.

References

[1] http://pdadb.net/index.php?m=cpu&id=a8960dt&c=qualcomm_snapdragon_s4_pro_msm8960dt