API Comparison of CPU-To-GPU Command Offloading Latency on Embedded Platforms (Artifact)

Roberto Cavicchioli
Università di Modena e Reggio Emilia, Italy
roberto.cavicchioli@unimore.it

Nicola Capodieci
Università di Modena e Reggio Emilia, Italy
nicola.capodieci@unimore.it

Marco Solieri
Università di Modena e Reggio Emilia, Italy
ms@xt3.it

Marko Bertogna
Università di Modena e Reggio Emilia, Italy
marko.bertogna@unimore.it

Abstract

High-performance heterogeneous embedded platforms allow offloading of parallel workloads to an integrated accelerator, such as General Purpose-Graphic Processing Units (GP-GPUs). A time-predictable characterization of task submission is a must in real-time applications. We provide a profiler of the time spent by the CPU for submitting stereotypical GP-GPU workload shaped as a Deep Neural Network of parameterized complexity. The submission is performed using the latest API available: NVIDIA CUDA, including its various techniques, and Vulkan. Complete automation for the test on Jetson Xavier is also provided by scripts that install software dependencies, run the experiments, and collect results in a PDF report.

2012 ACM Subject Classification Computer systems organization → System on a chip; Computer systems organization → Real-time system architecture

Keywords and phrases GPU, Applications, Heterogeneus systems

Digital Object Identifier 10.4230/DARTS.5.1.4

Related Article Roberto Cavicchioli, Nicola Capodieci, Marco Solieri, and Marko Bertogna, "Novel Methodologies for Predictable CPU-To-GPU Command Offloading", in 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), LIPIcs, Vol. 133, pp. 22:1–22:22, 2019.
https://dx.doi.org/10.4230/LIPIcs.ECRTS.2019.22

Related Conference 31st Euromicro Conference on Real-Time Systems (ECRTS 2019), July 9–12, 2019, Stuttgart, Germany

1 Scope

This artifact allows a characterization of the CPU-to-GPU submission latencies for real-time systems executing on heterogeneous embedded platforms. The experiments enable comparing the recently released CUDA submission models, and the novel open standard Vulkan API, profiling GPU command submission times and total execution times. Inferencing on a neural network of parameterized topology has been selected as a typical workload.

Obtainable results firstly show that CPU offloading latencies can act as a bottleneck for performance, negatively impacting predictability and jitter, and making the schedulability analysis significantly more complex, due to the large number of CPU-GPU interactions.

Secondly, considering CUDA approaches, the recently introduced submission models appear to slightly improve performance (both on submission and execution times) compared to the commonly utilized baseline approach. However, considering a deeper neural network and buffer data size, the performance penalties during the actual kernel computations reduce or invalidate the benefits of a reduced CPU activity gained for submission operations, especially for the CDP approach.

Finally, results show that the Vulkan API is able to minimize and better distribute the CPU overhead in all the tested configurations. This led to significant improvements, i.e. up to 11× faster submissions and from 2× to 6× faster GPU operations, with almost negligible jitter.

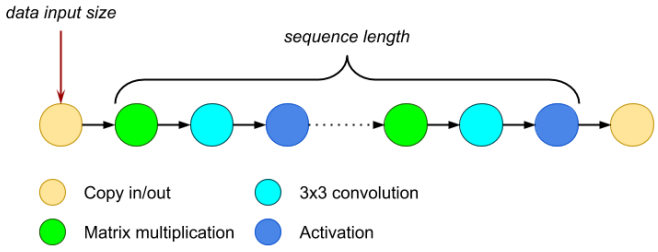
Further discussions are reported in the companion paper.

2 Content

The artifact package includes:

- `run_artifact.sh`: Provides the main test automation.
  1. Setup the board with highest supported frequencies and voltage for CPU and GPU, disabling dynamic scaling.
  2. Enable the scheduling parameter for launching a process with real time priority FIFO99.
  3. Install the Vulkan SDK.
  4. Run the test process, pinned on a single core.
  5. Aggregate and pre-process the results in CSV form.
  6. Install the minimum LaTeX environment to chart results.
  7. Compile and open the PDF report with charts.

  Steps 1-4 all are mandatory to reproduce results. If you do not want some of these steps executed (e.g., to follow a different installation method for some of the requirements, or to exclude plotting), corresponding lines can just be commented out from the script.
- `install_vulkan.sh`: Downloads the Vulkan SDK from `https://github.com/KhronosGroup/Vulkan-LoaderAndValidationLayers.git`, installs the related software dependencies from L4T (Ubuntu Xenial) distribution, and then build and install Vulkan.
- `tests.sh`: Launches four kinds of experiments, one for each GPU submission kinds: baseline, CUDA CDP, CUDA graphs, Vulkan. In each case, a deep-neural-network-like task graphs like the following is submitted:



  Input/output size matrices are $(k * block) \times (k * block)$, with $k \in [1, 2, 4, 8]$ and $block = 16$. Kernel launches length is given by $3 \times l$ with $l \in [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000]$. Each configuration is executed 500 times, measuring two response times: submission completion time, and total execution time. Further details are reported in the companion paper.
- `mainfile.cu`, `kernels.cu`, `vkcomp/stdafx.cpp`: CUDA C implementation of the test.
- `vkcomp/`: Vulkan implementation of the test.
- `table_jitter.py`: data aggregation script.
- `results.tex`, `lipics-v2019.cls`: Results reporting in PDF with LaTeX.

3 Getting the artifact

The artifact endorsed by the Artifact Evaluation Committee is available free of charge on the Dagstuhl Research Online Publication Server (DROPS). In addition, the artifact is also available at: `https://git.hipert.unimore.it/rcavicchioli/cpu_gpu_submission`.

4 Tested platforms

The artifact is known to work on NVIDIA Jetson AGX Xavier, on Jetpack 4.1.1 Developer Preview, including Linux4Tegra r31.

5 License

The artifact is available under the Simple Non Code License (SNCL) Version 2.1.0, whose verbatim text is included.

6 MD5 sum of the artifact

3978b2398eab0687e51009e681c0ada9

7 Size of the artifact

37961 bytes