TVM for Adreno GPUs

Qualcomm Technologies, Inc.

Li He, Hongqiang Wang,
Adarsh Golikeri and Alex Bourd
lih, wangh, adarshg, abourd@qti.qualcomm.com

Graphics Research Team, Qualcomm Technologies, Inc.

Introduction

TVM (Tensor Virtual Machine) [1] is a machine learning based compiler stack to automatically optimize the performance of deep learning networks through auto code generation, auto kernel fusion and autotuning. It can greatly facilitate performance optimization and portability, two major issues for machine learning network deployment.

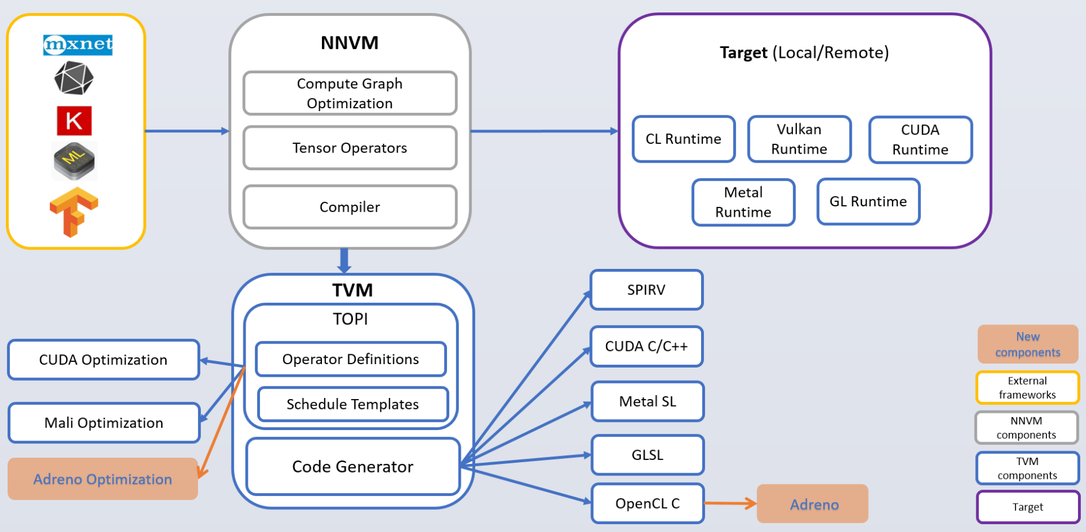
In this work, we add Qualcomm's advanced Adreno™ mobile GPU as a new target for TVM, which currently is not fully supported in the stack. Code generator, runtime support and autotuning templates have been updated for Adreno. We also improve the support of local memory by adding different caching strategies. In addition, support of one-dimensional image object (image1D) is added to TVM in order to leverage high throughput of the texture processor in Adreno GPUs and the auto boundary handling with image object. We use MobileNetV1 as the target network and apply the modified TVM stack on Adreno A640 GPU in Snapdragon 855 SOC and have achieved a speedup of 2.65x over the official TVM.

Why TVM for Adreno

- OpenCL performance is dependent on hardware and hardly portable across hardware generations/tiers.
- We developed many kernels for the most popular networks, e.g. MobileNetV1, InceptionV3, etc., but only cover the premium tier GPUs. Tuning for lower tier GPUs is time and cost prohibitive.
- Adreno GPUs have features and characteristics that are not expressed in TVM.
- TVM has been gaining tremendous traction in industry recently.

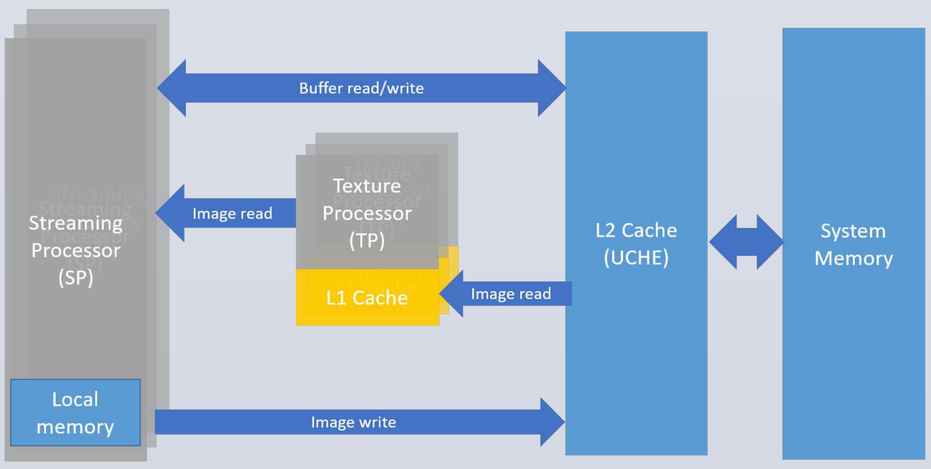
TVM Architecture

- The following diagram shows the architecture of TVM with Adreno-specific components introduced in this work.



Adreno Hierarchy

- The following diagram shows the high-level architecture of Adreno GPUs [2].
- Each streaming processor has on-chip local memory.
- Each texture processor has L1 cache.



TVM's Limitation and Challenges

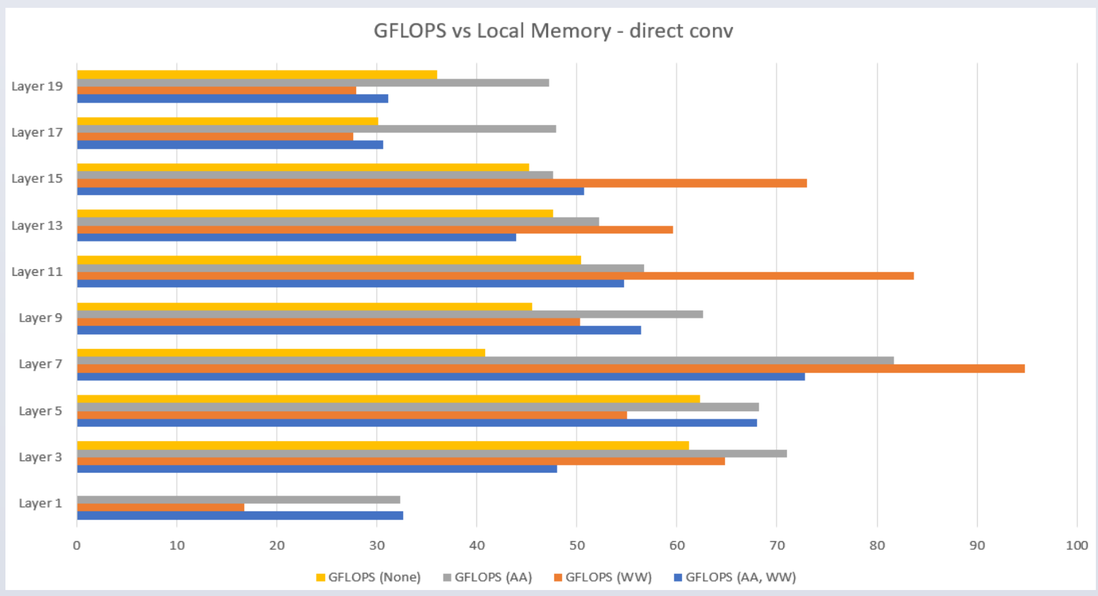
- TVM does not fully utilize local memory for mobile GPU targets. The OpenCL in TVM for mobile GPUs assumes there is no physical on-chip local memory, and therefore avoid using it (see sec. 7.3.2 in [3])
- The texture processor in Adreno GPUs is very powerful. However, it can be accessible only through image objects in OpenCL. However, TVM only supports buffers.
- We need to make TVM code generator and runtime be aware of Adreno features and expose them properly throughout the stack.
- Adding image object support requires changes in code generator, optimizing passes, runtime and autotuning templates.

Summary of Results

Strategy	Speedup	Execution time (ms)
Vanilla TVM only OpenCL buffers (baseline)	1.00	49.08
OpenCL buffer + local memory:	1.67	29.22
Image1d buffer (CL_R) + local memory	1.83	26.72
Image1d buffer (CL_R) + local memory + single layer finetuned	1.93	25.42
Image1d buffer (CL_R) + optimal local memory strategies	2.65	18.48
Hand optimized kernels, not TVM	**4.02**	**12.00**

Results and Discussions

- The optimal solution is obtained by combining different local memory strategy with use of image objects.
- We study four local memory modes:
  - AA – storing input activation in local memory
  - WW – storing weights in local memory
  - AA,WW – storing activation and weights in local memory
  - None – do not use local memory
- This is also applied to depth-wise 2D convolution.
- AutoTVM has been enabled to select the optimal local memory strategy:
  - The optimal one: Image1d buffer + local memory strategies
  - 2.65x performance boost



	Input path	Weight path
AA	TP -> LM	TP
WW	TP	TP -> LM
AA,WW	TP -> LM	TP -> LM
None	TP	TP

Conclusions and Future Work

By utilizing the on-chip local memory and 1D image object, we were able to improve the performance of stock TVM on Adreno GPU A640 by ~2.65x. Though still behind the hand optimized version, the modified TVM for Adreno shows its great potential. Note that many features that are performance critical for Adreno GPUs have not been added to TVM yet. For example, we only use 1D image with CL_R format, which is far from optimal as compared with 2D image objects with CL_RGBA format. Adding support of 2D image object and CL_RGBA is not as straightforward as it looks, due to the challenges in address calculation, handling of out of bound condition, and the access pattern limit (e.g., not byte addressable). In addition to 2D image, other features like subgroup and better use of constant memory could be added in TVM in future.

References

[1] Tianqi Chen, et al, TVM: An Automated End-to-End Optimizing Compiler for Deep Learning, OSDI 2018.

[2] Qualcomm Technologies, Inc., Qualcomm® Snapdragon™ Mobile Platform OpenCL General Programming and Optimization (80-NB295-11 A), 2017. https://developer.qualcomm.com/qfile/33472/80-nb295-11_a.pdf

[3] ARM Limited, Mali™ GPU OpenCL Developer Guide v3.2, 2016.

IWOCL '20, April 27–29, 2020, Munich, Germany

ACM ISBN 978-1-4503-7531-3/20/04.
https://doi.org/10.1145/3388333.3388667