Critique on TVM, an Automated End-to-End Optimizer Compiler

Summary

The TVM compiler provides a way of optimizing DL Frameworks on a given hardware specification. Rather than rely on a high-level graph-based optimization that existing software frameworks do, it is able to generate low-level optimized code from the high-level specification of existing frameworks such as TensorFlow, Pytorch, Keras etc.. It is able to do this for a wide variety of hardware specifications like GPUs, embedded CPUs, and generic FPGA-based accelerators. Moreover, TVM optimizes the mappings through an ML-based approach, using tensor expression language to build operators and automatically exploring optimizations. TVM achieves this through generating many implementations of each hardware backend, and then choosing the one that is most optimized. TVM also has features such as nested-parallelism, tensorization and explicit memory latency hiding. An ML-based cost model is used to find the best schedule, and it is shown to perform better that simplistic algorithmic models.

Strengths

- Provides automated optimazation for existing frameworks, requiring no additional effort in mapping to compatible language
- A large variety of HW can be specified and optimized for.

Weaknesses

- Focuses on mapping for a specific configuration. Does not suggest improvements on the hardware model itself that could make the inference faster
- Only generic hardware models can be described, something that has a very novel NoC can be hard to model just through latencies in read and write

Suggested Improvements

- Possible inspiration from papers such as timeloop that would enable a better search over possible HW configurations