TVM at Qualcomm Technologies

Krzysztof Parzyszek
Qualcomm Innovation Center, Inc.
Goals

• Add support for Qualcomm® Hexagon™ architecture to TVM
  • Low power processor with DSP features (32 32-bit registers, vector operations on 64-bit register pairs)
  • HVX coprocessor with 128-byte vectors (32 1024-bit registers)
  • Rich instruction set (both core and HVX) for efficient integer numerical computations
  • Well suited for integer ML workloads

• Make TVM fully exploit capabilities of Hexagon processors
  • Instruction complexity makes it hard for C/C++ compilers to generate them
  • Having a domain-specific programming environment makes it easier to utilize complex hardware features

• Make TVM available to our customers for all available Hexagon chips
  • Including scalar cores and those with HVX
  • Experience with Halide shows that having a DSL is very useful for our customers
Current status

- Internal releases of TVM available within Qualcomm Technologies, Inc.
- Collecting information for future development plans
  - Feedback from users
  - Performance measurements
- Developing kernels to expand our ML offerings
What we have done so far

- Support all scheduling directives: parallel, prefetch, vectorize, etc.
- Hexagon-specific schedules for a set of operations
- Use `bind.thread_axis("pipeline")` to mark a kernel for offloading, not needed for standalone ops
- Basic LLVM-based codegen to implement necessary functionality
- Runtime for enabling execution on Hexagon simulator
  - Execute Hexagon kernels on a PC
  - Runtime for Android enabling execution on Hexagon hardware
  - Includes FastRPC communication layer
- It's a cat.
What we have done so far: technicalities

• Engineering notes
  • Complex HVX instructions can be generated via tensorization, but not automatically
  • Simulator only supports single thread
  • HVX code generated via “vectorize”, or via auto-vectorization in LLVM, scalar vectors only via “vectorize”
  • Support Hexagon V62 and later, scalar cores and HVX
  • Use `tvm.target.hexagon(...)` to select build target, parameters allow CPU selection, simulator configuration, LLVM options
  • Device runtime supports ADSP and CDSP

• Performance
  • Comparing with hand-written assembly: favorable results for simple ops on small data sets
Special use case: ML op compiler

- Use TVM to compile ops for other applications
- Such ops will execute without TVM runtime
  - Create a shim runtime that implements TVM/RT API on top of another runtime
- Such ops may also need to meet additional requirements
  - The other ML framework’s runtime may require extra information be passed from the calling op
- Qualcomm AI Research team will present a demo at NeurIPS
Challenges

• Prefetch
  • Hexagon has a “rectangular” prefetch (l2fetch), while TVM assumes a cache-line prefetch
  • Invented “prefetch” intrinsic, delayed expansion from storage flattener until TVM intrinsic expansion
  • Will need to work with the community on integrating this into TVM sources

• Propagating buffer alignment information
  • Storage alignment needs to be passed to compute functions and to outlined device functions

• Storage control
  • Stack limit and default alignment are hardcoded; HVX needs 128-byte alignment, but scalar code does not,
    Hexagon stack can be much larger than the limit

• Need to add custom attributes
  • Some internal use cases require special parameters be passed to ops, we invented an attribute to annotate
    such parameters

• Routine application of specific lowering passes
Future work

• Support compilation of Relay directly to Hexagon code
  • Build entire graphs and subgraphs for Hexagon

• Extend code generation to exploit more HVX instructions and features

• Upstream
  • Make our backend available in the public repository
  • Publish bug fixes
  • Contribute our extensions and enhancements to the target-independent code

• We have had a very positive experience with TVM
  • Using TVM as a ML compiler for Hexagon shows a lot of potential

• Hope to work closely with TVM community