Supporting TVM on RISC-V Architectures with SIMD Computations

Jenq-Kuen Lee¹, Chun-Chieh Yang¹, Allen Lu², P. Chen¹, YM Chang¹,², CH Chang¹, Yi-Ru Chen¹, HH Liao¹, Chao-Lin Lee¹,², Ssu-Hsuan Lu², and Shao-Chung Wang³

¹Department of Computer Science, National Tsing Hua University, Taiwan
²Peakhills Group Corporation
³Andes Technology Corporation
RISC-V with two vector ISAs to support fall-back engine with AI Models

**Super Word Vector V Extension**

```
| v0 | e7 e6 e5 e4 e3 e2 e1 e0 |
| v1 | e7 e6 e5 e4 e3 e2 e1 e0 |
| v2 | e7 e6 e5 e4 e3 e2 e1 e0 |
| v3 | e7 e6 e5 e4 e3 e2 e1 e0 |
| v28| e7 e6 e5 e4 e3 e2 e1 e0 |
| v29| e7 e6 e5 e4 e3 e2 e1 e0 |
| v30| e7 e6 e5 e4 e3 e2 e1 e0 |
| v31| e7 e6 e5 e4 e3 e2 e1 e0 |
```

**Packed Vector (SubWord SIMD) P Extension With Fixed-Point and Integer Instructions**

- Add
- Sub
- Mul
- Compare
- Signed
- Unsigned

RISC-V DSP (P) Extension Proposal Chuan-Hua Chang, Andes Technology Corporation

Courtesy: Vector ISA, Roger Espasa, Esperanto Technologies
[RFC] Fixed-point type implementation proposal #4446

- RISC-V P extension (Subword SIMD) with fixed-point instructions.
- We refer Fxp as fixed-point value, Fp as floating-point value and PP as point position
  - Fxp = Fp * pow(2,PP)
- Support Fixed-point Type with TVM
- Compiler time with type information for the binary point position of the variable.

References for Fixed-Point Type
(1) AC fixed-Point by Mentor graphics (https://www.mentor.com/hls-lp/downloads/ac-datatypes)
Auto-FXP with TVM on RISC-V with p Extension

- Using machine learning model to auto-tune the binary point position.
- It can find the best binary point position for fixed-point expression when we have TVM on RISC-V with p extension.
- The work extends AutoTVM and can enhance the accuracy while enjoy the low power numeric benefits.
- The tuning work is done with spike simulator incorporated with RISC-V P extension (Subword SIMD).
TVM for RISC-V with V Extension (Superword SIMD)

- TVM Optimization
  - The TVM RISC-V codegen will lower SIMD computation with SIMD intrinsics into LLVM.
  - The LLVM backend will need to generate the corresponding SIMD instructions.
  - Need to tune the scheduler to provide a large loop index space for vector parallelism.

- LLVM Optimization
  - VSETVL Redundancy Elimination
  - VMulADD Resource Utilization
  - Fast Vector Initializer

- Spike Simulator
  - Assume 512 bits vector register
  - V SIMD in <4 x float32>, <8 x float32>, <16 x float32>
  - Spec v0.7.0, TVM v0.6, LLVM 9.0.0
  - Compare with SIMD float32 and no SIMD float32

Speedup based on runtime executed instructions

<table>
<thead>
<tr>
<th></th>
<th>Only TVM Optimization</th>
<th>TVM+ LLVM Optimization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenet</td>
<td>2.58</td>
<td>4.69</td>
</tr>
<tr>
<td>AlexNet</td>
<td>2.72</td>
<td>5.02</td>
</tr>
<tr>
<td>SqueezeNet1.0</td>
<td>2.28</td>
<td>4.44</td>
</tr>
<tr>
<td>Mobilenetv2</td>
<td>2.74</td>
<td>4.98</td>
</tr>
<tr>
<td>Resnet18_v1</td>
<td>3.83</td>
<td>8.77</td>
</tr>
<tr>
<td>Densenet</td>
<td>2.95</td>
<td>5.60</td>
</tr>
<tr>
<td>inceptionv3</td>
<td>2.19</td>
<td>4.36</td>
</tr>
<tr>
<td>AVG</td>
<td>2.49</td>
<td>4.67</td>
</tr>
</tbody>
</table>
Summary

- Thank you AWS team help with AI model validation flow.
- Look forward to contributing codes to the TVM source trees.
- More detailed of our work can also be found in the following.