Marvel: A Heuristic-based Mapper for Mapping Space Exploration of DNN Operators onto Spatial Accelerators Using MAESTRO

Prasanth Chatarasi

MAESTRO Tutorial - MICRO 2020
October 17th, 2020

"Marvel: A Data-centric Compiler for DNN Operators on Spatial Accelerators"
Prasanth Chatarasi, Hyoukjun Kwon, Natesh Raina, Saurabh Malik, Vaisakh Haridas, Angshuman Parashar, Michael Pellauer, Tushar Krishna, and Vivek Sarkar (ArXiv’20)
Deep Learning (DNN Models)

Examples of DNN Operators (Layers)
- Regular CONV1D
- Regular CONV2D
- Depth-wise CONV2D
- Transposed CONV2D
- Regular CONV3D
- Strided variants
- GEMM (MatMul)
- LSTM (RNNs)
- Element-wise
- Pooling
- Fully Connected/MLP
- …..

Regular CONV2D over 4D Tensors

Involves billions of computations

Parashar et al., ISPASS 2019
Mapping problem

DNN Operators
- Regular CONV1D
- Regular CONV2D
- Depth-wise CONV2D
- Transposed CONV2D
- Regular CONV3D
- Strided variants
- GEMM (MatMul)
- LSTM (RNNs)
- Element-wise
- Pooling
- Fully Connected/MLP
- …..

Mapping involves
1) Parallelization onto compute resources,
2) Tiling across memory resources, and
3) Exploitation of data reuse

Abstract overview

How to map for low latency, high energy efficiency?

3-level accelerator
E.g., TPU, Eyeriss, NVDLA
Challenges

1. Explosion of hardware choices in spatial accelerators
   • Wide variety of hardware structures & data movement restrictions

2. Rapid emergence of new DNN operators and shapes/sizes
   • Various forms of algorithmic properties (e.g., reuses)

3. Selection of optimized mapping from massive mapping space and also good cost models
   • E.g., On average, $O(10^{18})$ mappings for CONV2D in MobileNetV2

"Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach"
Hyoukjun Kwon, Prasanth Chatarasi, Michael Pellauer, Angshuman Parashar, Vivek Sarkar, and Tushar Krishna,
In Proceedings of the 52nd IEEE/ACM International Symposium on Microarchitecture (MICRO’19)
Mapping space for a 3-level accelerator

- Multi-level tiling for memory hierarchy and for parallelization
  1. Level-1 tiling for the L1 buffer
  2. Level-2 tiling for the PE array
  3. Level-3 tiling for the L2 buffer

- Loop orders across tiles
  4. Inter-tile level-3 loop order
  5. Inter-tile level-2 loop order

- Data-layouts
  6. Tensors on DRAM

- Mapping is an unique 6D tuple in the 6-dimensional search space

$O(10^{18})$ mappings on average for a single convolution layer in ResNet50 and MobileNetV2 models on Eyeriss-like accelerator
Our Intuition

Observation: Off-chip data movement is 2-3 orders of magnitude more expensive compared to on-chip data movement

Idea: Decouple the mapping space based on off-chip and on-chip data movement, and prioritize optimizing for off-chip data movement first?

Vivienne et al., Deep Learning Tutorial
Our approach (Marvel)

Mapping space (6-dimensional)

- Off-chip Subspace
  - (3-dimensional)
    - Tensor Data Layout
    - Lv-3 Tile Size
    - Lv-3 Tile Order
    - Decoupling Heuristic
    - Optimal data layout
    - Optimal level-3 tile sizes
    - Optimal level-3 tile order

- On-chip Subspace
  - (3-dimensional)
    - Tensor Data Layout
    - Lv-2 Tile Size
    - Lv-2 Tile Order
    - Optimal level-2 tile sizes
    - Optimal level-2 tile order

Decoupled mapping space

- Off-chip subspace
- On-chip subspace

Cost models

- Distinct Blocks (DB) Cost Model
  - Sarkar et al., IBM Journal, 1997

- MAESTRO Cost Model
  - Kwon et al., MICRO 2019

Optimal mapping

- Optimal level-2 tile orders
- Optimal level-2 tile sizes
- Optimal level-1 tile sizes
Step-1: Optimizing off-chip subspace

- **Input:** Workload, hardware configuration, and exploration options
- **Output:** Level-3 tile sizes & inter-tile order, and data-layouts

**Distinct Blocks Model (DB Model)**
- Given a parametric loop-nest and layout of tensors, the model measures distinct number of DRAM blocks for a computation tile

```plaintext
for(n=0; n<N; n++)
  for(k=0; k<K; k++)
    for(c=0; c<C; c++)
      for(p=0; p<P; p++)
        for(q=0; q<Q; q++)
          for(r=0; r<R; r++)
            for(s=0; s<S; s++)
              O[n][k][r][p] += W[k][c][r][s] * I[n][c][q+r][p+s];
```

\[ T_{3i} \text{ is the tile size for loop-i,} \]
\[ b \text{ is the DRAM block size} \]

\[
DB_W(T_3) \approx \left( \left\lfloor \frac{T_{3S}}{b} \right\rfloor \right) \times T_{3R} \times T_{3C} \times T_{3K}
\]
\[
DB_I(T_3) \approx \left( \left\lfloor \frac{T_{3P} + T_{3S}}{b} \right\rfloor \right) \times (T_{3Q} + T_{3R}) \times T_{3C} \times T_{3N}
\]
\[
DB_O(T_3) \approx \left( \left\lfloor \frac{T_{3P}}{b} \right\rfloor \right) \times T_{3Q} \times T_{3K} \times T_{3N}
\]

\[ DMC(T_3) \approx \frac{DB_W(T_3) + DB_I(T_3) + DB_I(T_3)}{T_{3N} \times T_{3K} \times T_{3C} \times T_{3X} \times T_{3Y} \times T_{3R} \times T_{3S}} \]

\[ b \times DB_{Total}(T_3) \leq \frac{|L_2|}{2} \]

Sarkar et al., IBM Journal, 1997
Step-2: Optimizing on-chip subspace

- **Input:** Level-3 tile sizes, Level-3 tile order, data-layouts
- **Output:** Level-2 tile sizes, Level-2 tile order, Level-1 tile sizes
- Iterate over each on-chip mapping, translate into MAESTRO understandable format, and invoke MAESTRO cost model

A) CONV1D operation

```plaintext
for(i = 0; i < M; i++)
  for(j = 0; j < N; j++)
    O[i] += I[i+j] * W[j]
```

B) DDG

C) A sample mapping in the loop-nest representation

Level-1 tile sizes: $T_{1i}, T_{1j}$
Level-2 tile sizes: $T_{2i}, T_{2j}$
Level-2 inter-tile order: $t_{3i}, t_{3j}$

D) Mapping directives

```plaintext
#pragma omp parallel for
for(t_{3i} = t_{4i}; t_{3i} < t_{4i}+T_{3i}; t_{3i}+=T_{2i})
  for(t_{3j} = t_{4j}; t_{3j} < t_{4j}+T_{3j}; t_{3j}+=T_{2j})
    O[t_{1i}] += I[t_{1i}+t_{1j}] * W[t_{1j}]
```
Impact of our approach over CONV2D mapping space in Modern DNN models

<table>
<thead>
<tr>
<th>Variants</th>
<th>Search space size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Min</td>
</tr>
<tr>
<td>Original search space</td>
<td>$2.7 \times 10^{17}$</td>
</tr>
<tr>
<td>Off-chip schedules search space after decoupling</td>
<td>$7.3 \times 10^{8}$</td>
</tr>
<tr>
<td>On-chip schedules search space after decoupling</td>
<td>$2.9 \times 10^{7}$</td>
</tr>
<tr>
<td>Off-chip schedules search space after decoupling + pruning</td>
<td>$9.9 \times 10^{5}$</td>
</tr>
<tr>
<td>On-chip schedules search space after decoupling + pruning</td>
<td>$3.8 \times 10^{5}$</td>
</tr>
</tbody>
</table>

TABLE IV: The statistics (min/avg/max) of the mapping space of convolution layers in our evaluation and the resultant mapping subspaces after decoupling and pruning strategies.

Because of our decoupled and pruning strategies, on an average, mapping space is reduced by $O(10^{10})$
Demo of Marvel

- **HW: 3-level edge class accelerator**
  - PEs = 1024, NoC (bi-directional) bandwidth = 25.6 GBps, L1 size = 512 Bytes, and L2 size = 108KB

- **Layer1: CONV2_2_2 (REGULAR CONV2D) from VGG16**
  - Mappings different for optimal latency, energy, and edp
  - Varying number of levels of parallelism

- **Layer2: Bottleneck6_2_2 (Depth-wise CONV2D) from MobileNetV2**
  - Mappings different for optimal latency, energy, and edp
  - Performance limited by NoC bandwidth
Summary and in-progress works

1. Marvel — Decoupled off-chip/on-chip heuristic to efficiently explore the massive search space of mappings
   • Reduced the search space on an average by \( O(10^{10}) \)

2. Existing exploration strategies in Marvel framework
   • Off-chip/on-chip decoupling approach
   • Brute-force, Random Sampling,
   • Prior strategies for CONV2D such as Interstellar, dMazeRunner

3. In-progress work
   • Integration with MLIR framework
   • Integration with RL-based exploration strategies
   • Support for more DNN operators
   • Source code will be released soon!

Overall, we view Marvel as a research infrastructure to help compiler and micro-architecture research of accelerators!