CoSA: Scheduling by Constrained Optimization for Spatial Accelerators

Qijing Huang, Minwoo Kang, Grace Dinh, Thomas Norell, Aravind Kalaiah†, James Demmel, John Wawrzynek, Yakun Sophia Shao

UC Berkeley, †Facebook
Scheduling is required everywhere

- Algorithm
  - algorithmic states to be run

- Hardware
  - hardware resources to be allocated

Motivation
Scheduling is a big challenge

- Algorithm

1. Exponentially growing algorithm complexity
Exponentially growing algorithm complexity

Model Complexity (# of parameters)

DNN model size doubles every 3.5 months

* source from Intel AI
Scheduling is a big challenge

1. Exponentially growing algorithm complexity
2. Rapidly increasing hardware capacity

- Algorithm
- Hardware
Rapidly increasing hardware capacity

NoC/NoP Chip

(a) Simba chiplet  
(b) Simba package

Simba¹
16PEs x 36 Chiplets

Wafer-scale Chip

Cerebras²
84 Interconnected Chips

Scheduling is a big challenge

1. Exponentially growing algorithm complexity
2. Rapidly increasing hardware capacity
Scheduling significantly affects performance

![Graph showing latency vs. number of valid schedules with a 7.2x increase]
State-of-the-art DNN accelerator schedulers

**Brute-force**
- Timeloop
- dMazeRunner
- Triton
- Interstellar
- Marvel

**Feedback-based**
- AutoTVM
- Halide
- FlexFlow
- Gamma
- MindMapping

**Constrained Optimization**
- Polly+Pluto
- TC
- Tiramisu
- CoSA

- Costly
- Sample invalid space
- Hard to generalize

- Unable to determine tiling factor sizes

One-shot solution
Opportunities

- Workload Regularity
- Hardware Regularity
- Explicit Data Movement
Target Workload

\[ \text{Inputs (IA)} \quad \text{Weights (W)} \quad \text{Outputs} \]

\[ (P - 1) \times \text{Stride} + R \quad (Q - 1) \times \text{Stride} + S \]

\( R, S \): weight width and height
\( P, Q \): output width and height
\( C \): input channel size
\( K \): output channel size
\( N \): batch size

DNN Layer:

\[ \text{for } n \text{ in } [0:N) \]
\[ \quad \text{for } k \text{ in } [0:K) \]
\[ \quad \text{for } c \text{ in } [0:C) \]
\[ \quad \text{for } p \text{ in } [0:P) \]
\[ \quad \text{for } q \text{ in } [0:Q) \]
\[ \quad \text{for } r \text{ in } [0:R) \]
\[ \quad \text{for } s \text{ in } [0:S) \]
\[ \text{OA}[n,p,q,k] += \]
\[ \text{IA}[n,p+r-(R-1)/2,q+s-(S-1)/2,c] \times W[r,s,c,k] \]
Target Architecture

- Spatial PEs
- Multi-level Memory Hierarchy

<table>
<thead>
<tr>
<th>Buffer Type</th>
<th>Dimensions</th>
<th>Entries</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Registers</td>
<td>8x8</td>
<td>1</td>
<td>64B</td>
</tr>
<tr>
<td>AccumBuffer</td>
<td>1x8</td>
<td>128</td>
<td>384B</td>
</tr>
<tr>
<td>WeightBuffer</td>
<td>1x8</td>
<td>4096</td>
<td>4KB</td>
</tr>
<tr>
<td>InputBuffer</td>
<td>1x1</td>
<td>8192</td>
<td>8KB</td>
</tr>
</tbody>
</table>

64 MAC

DNN Accelerator
DNN scheduling problem formulation with CoSA

\[(P - 1) \times \text{Stride} + R \times (Q - 1) \times \text{Stride} + S\]

- **R, S**: weight width and height
- **P, Q**: output width and height
- **C**: input channel size
- **K**: output channel size
- **N**: batch size
Three scheduling decisions

1. Tiling Factors

2. Spatial / Temporal

3. Loop Permutation

**DRAM level**

```plaintext
for q2 = [0 : 2) :
```

**Global Buffer level**

```plaintext
for q1 = [0 : 7) :
  for n0 = [0 : 3) :
    spatial_for r0 = [0 : 3) :
    spatial_for k1 = [0 : 2) :
```

**Input Buffer level**

```plaintext
for c1 = [0 : 2) :
  for p1 = [0 : 2) :
```

**Weight Buffer level**

```plaintext
for p0 = [0 : 2) :
  spatial_for k0 = [0 : 2) :
```

...
Key idea: prime factor allocation problem

Matrix-vector mult:
for c in [0:C) // C = 28
  for k in [0:K) // K = 15
    OA[k] += IA[c] × W[c,k]

Prime factor items:

Local buffers:
- Weight buffer
- Global buffer

Weight Buffer
(Size = 4)

Global Buffer
(Size = 20)
CoSA Variable X – Tiling Factors

Prime factor items:

- C = 28
  - Prime Factors: 2
  - WeightBuf ✓
  - GlobalBuf ✓ ✓ ✓
  - DRAM ✓

- K = 15
  - Prime Factors: 3, 5

Local buffers:

- WeightBuffer (Size = 4) Utilized: 2
  - Utilized: (2\times3\times5)\times2=60
- GlobalBuffer (Size = 80)

Binary allocation var X:

<table>
<thead>
<tr>
<th>Prime Factors</th>
<th>C=28</th>
<th>K=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>WeightBuf</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>GlobalBuf</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>DRAM</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Utilized: 2
CoSA Variable X – Spatial/Temporal Mapping

Prime factor items:

<table>
<thead>
<tr>
<th></th>
<th>C = 28</th>
<th>K = 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Spatial: ✓
Temporal: ✓

4 PEs in the accelerator:

Spatial Factors (Limit=4)

Global Buf (Size = 80)

Global Buffer

Binary allocation var X:

<table>
<thead>
<tr>
<th>Global Buf</th>
<th>C=28</th>
<th>K=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime Factors</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Spatial</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Temporal</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
CoSA Variable X – Loop Permutation

Prime factor items:

<table>
<thead>
<tr>
<th>C = 28</th>
<th>K = 15</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Prime Factor Items for C=28" /></td>
<td><img src="image2" alt="Prime Factor Items for K=15" /></td>
</tr>
</tbody>
</table>

Prime Factors:

- C=28: 2, 2, 7
- K=15: 3, 5

Rank in global buf:

- rank0: ✓
- rank1: ✓
- rank2: 
- rank3: 
- rank4: 

Binary allocation var X:

<table>
<thead>
<tr>
<th></th>
<th>C=28</th>
<th>K=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prime Factors</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>rank0</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>rank1</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>rank2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rank3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rank4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Global Buffer (Size = 80)
### CoSA Variable X – Putting it altogether

<table>
<thead>
<tr>
<th>Prime Factors</th>
<th>Perm</th>
<th>C=28</th>
<th>K=15</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td></td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Perm</td>
<td></td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

| WeightBuf | ... | t |
| GlobalBuf | rank0 | t |
|          | rank1 |   |
|          | rank2 |   |
|          | rank3 |   |
|          | rank4 |   |
| DRAM     | ...  | t |

**DRAM level**

\[
\text{for } c2 = [0 : 7) : \\
\]

**Global Buffer level**

\[
\text{for } k1 = [0 : 5) : \\
\text{for } c1 = [0 : 2) : \\
\text{for } k0 = [0 : 3) : \\
\]

**Weight Buffer level**

\[
\text{for } c0 = [0 : 2) : \\
\]

s - Spatial, t - Temporal
CoSA Constraints: Buffer Utilization

Weight Buffer (Size = 4)

Weight Buffer (Size = 4)

Weight Buffer (Size = 4)
CoSA Constraints: Spatial Resources

Spatial PEs (Limit = 4)

Spatial PEs (Limit = 4)

Spatial PEs (Limit = 4)
CoSA Objectives

- Utilization-driven
- Compute-driven
- Traffic-driven
CoSA Objectives

- Utilization-driven
- Compute-driven
- Traffic-driven
CoSA Objectives

- Utilization-driven
- Compute-driven
- Traffic-driven
CoSA Objectives

- Utilization-driven
- Compute-driven
- Traffic-driven
CoSA Traffic-driven Objective

\[
\text{Overall Traffic} = S \times L \times D
\]
CoSA Evaluation

• Baselines:
  • Random (best out of 5 valid schedules)
  • Timeloop Hybrid (best out of 16K valid schedules)

• DNN workloads:
  • AlexNet, ResNet-50, ResNext-50, DeepBench

• Platforms:
  • Timeloop Simulator
  • SystemC NoC Simulator
  • GPU
1.5x latency speedup

- 5.2x better than Random
- 1.5x better than Timeloop Hybrid
1.2x better energy efficiency

- 3.3x better than Random
- 1.2x better than Timeloop Hybrid
90x faster time-to-solution with CoSA

<table>
<thead>
<tr>
<th></th>
<th>CoSA</th>
<th>Random</th>
<th>Timeloop Hybrid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime / Layer</td>
<td>4.2s</td>
<td>4.6s (1.1x)</td>
<td>379.9s (90.5x)</td>
</tr>
<tr>
<td>Samples / Layer</td>
<td>1</td>
<td>20K</td>
<td>67M</td>
</tr>
<tr>
<td>Evaluations / Layer</td>
<td>1</td>
<td>5</td>
<td>16K</td>
</tr>
</tbody>
</table>

- Generates schedules within seconds
- Significantly reduces the number of samples and evaluations
CoSA generalizes to different architectures

- Larger Buffers – 1.4x speedup
- 8x8 PEs – 1.1x speedup

- GPU – 1.2x speedup, 2500x faster time-to-solution over TVM (50 samples)
Conclusion

• We formulate DNN accelerator scheduling as a constrained optimization that can be solved in **one shot**.
• We take a *communication-oriented* approach in the formulation and exposes the cost through clearly-defined objective functions.
• We demonstrate that CoSA can **quickly** generate **high-performance** schedules outperforming state-of-the-art approaches.

Github: [https://github.com/ucb-bar/cosa](https://github.com/ucb-bar/cosa)

Questions?
qijing.huang@berkeley.edu