TENET: A Framework for Modeling Tensor Dataflow Based on Relation-centric Notation

Liqiang Lu¹, Naiqing Guan², Yuyue Wang¹, Liancheng Jia¹,
Zizhang Luo¹, Jieming Yin³, Jason Cong⁴, Yun Liang¹

¹ CECA, School of EECS, Peking University
² Computer Science Department, University of Toronto
³ Electrical and Computer Engineering, Lehigh University
⁴ Computer Science Department, University of California

Group URL: http://ceca.pku.edu.cn/
Email: liqianglu@pku.edu.cn, ericlyun@pku.edu.cn
Tensor applications

- Deep learning
  - CONV, GEMV, GEMM, GEMMc

- Image processing
  - CONV, Stencil, Jacobi-2D

- Recommendation system
  - GEMM, MTTKRP, TTMc

Tensor computation
Tensor computation

Various tensor size & different data reduction

Regular computation & huge computation size
Tensor-specific accelerators

Google TPU

Cambricon MLU270

Huawei Ascend

The key component is the computation dataflow
What is tensor computation dataflow?

Parallel computation

- Google TPU systolic dataflow
- Huawei Ascend 3D-dataflow
- Cambricon MLU270 multicast dataflow

Data movement
Various tensor dataflows

Which one has the lowest latency?
- Multicast
- Output stationary systolic
- Input stationary systolic

Which one has the lowest bandwidth requirement?
- Eyeriss Row Stationary
- Multicast+systolic
- Reduction Tree
We need a framework to analyze the dataflow

Dataflow notation

```
for (i = 0; i < 2; i++)
for (j = 0; j < 2; j++)
for (k = 0; k < 4; k++)
  S: Y[i,j] += A[i,k] * B[k,j];
```

Performance model

- Throughput
- Data reuse
- Data access
- Bandwidth
- Latency

Execution sequence:
- S[0,0,0]
- S[0,0,1]
- S[0,1,1]
Existing notations

Compute-centric notation:

do in pipeline

for (j = 0; j < 3; j++)
for (i = 0; i < 4; i++)
S: Y[i] += A[i+j]*B[j];

Data-centric notation:

for (j = 0; j < 3; j++)
for (i = 0; i < 4; i++)
S: Y[i] += A[i+j]*B[j];

Spatial map (1,1) i
Temporal map (1,1) j

less expressive
limited dataflow design space
less opportunities for optimization

Compute-centric notation reference:
Yang, Xuan, et al. "Interstellar: Using Halide’s Scheduling Language to Analyze DNN Accelerators." ASPLOS, 2020

Data-centric notation reference:
Design space of existing notations

Dataflow example

The time-stamp is an affine transformation of multiple loop dimensions

$T[i+j] \rightarrow A[i,j]$
Performance model of existing notations

Loop nest:
for (k = 0; k < 3; k++)
do in parallel:
   for (i = 0; i < 3; i++)
      for (j = 0; j < 3; j++)
         instance: $S[i, j, k]$
         $Y[i, j] += A[i, k] * B[k, j];$

Simple polynomials!

Integer set operators

Dataflow relation set

investigate each point in the dataflow set

more precise
Relation-centric notation overview

Relation: instance, data, PE, execution cycle

Dataflow relation

Data assignment relation

Interconnect relation

Precisely model the tensor dataflow on spatial architectures
Dataflow relation

Example: matrix multiplication

\[
\begin{array}{ccc}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{array}
\times
\begin{array}{ccc}
  A & B & C \\
  D & E & F \\
  G & H & I
\end{array} =
\begin{array}{ccc}
  1 & 2 & 3 \\
  4 & 5 & 6 \\
  7 & 8 & 9
\end{array}
\]

Dataflow 1:
Systolic array
output stationary

Dataflow 2:
Systolic array
input stationary

\[
\text{loop nest:}
\begin{align*}
  &\text{for } (i = 0; i < 3; i++) \\
  &\text{for } (j = 0; j < 3; j++) \\
  &\text{for } (k = 0; k < 3; k++) \\
  &\text{instance: } S[i,j,k]: \\
  &Y[i,j] += A[i,k] * B[k,j];
\end{align*}
\]

instance to PE
\{S[i,j,k] \rightarrow PE[i,j]\}

instance to cycle number
\{S[i,j,k] \rightarrow T[i+j+k]\}

affine transformation

instance to PE
\{S[i,j,k] \rightarrow PE[j,k]\}

instance to cycle number
\{S[i,j,k] \rightarrow T[i+k]\}

affine transformation
Step by step example

instance operation

\[ S[i,j,k] \quad A[i,k]*B[k,j] \]

\[
\begin{align*}
S[0,0,0] &: A[0,0]*B[0,0]; \\
S[0,0,1] &: A[0,1]*B[1,0]; \\
S[0,0,2] &: A[0,2]*B[2,0]; \\
S[0,0,3] &: A[0,3]*B[3,0]; \\
S[0,1,0] &: A[0,0]*B[0,1]; \\
S[0,1,1] &: A[0,1]*B[1,1]; \\
S[0,1,2] &: A[0,2]*B[2,1]; \\
S[0,1,3] &: A[0,3]*B[3,1]; \\
S[1,0,0] &: A[1,0]*B[0,0]; \\
S[1,0,1] &: A[1,1]*B[1,0]; \\
S[1,0,2] &: A[1,2]*B[2,0]; \\
S[1,0,3] &: A[1,3]*B[3,0]; \\
S[1,1,0] &: A[1,0]*B[0,1]; \\
S[1,1,1] &: A[1,1]*B[1,1]; \\
S[1,1,2] &: A[1,2]*B[2,1]; \\
S[1,1,3] &: A[1,3]*B[3,1];
\end{align*}
\]

Dataflow relation

\{ S[i,j,k] \rightarrow PE[i,j]\}  
obtain PE id

\{ S[i,j,k] \rightarrow T[i+j+k]\}  
obtain execution cycle
Data assignment relation

Tensor kernel tells:
- Tensor Y to instance
  \{Y[i,j] \rightarrow S[i,j,k]\}
- Tensor A/B to instance
  \{A[i,k] \rightarrow S[i,j,k]\}
  \{B[k,j] \rightarrow S[i,j,k]\}

Dataflow relation tells:
- Instance to PE
  \{S[i,j,k] \rightarrow PE[i,j]\}
- Instance to cycle number
  \{S[i,j,k] \rightarrow T[i+j+k]\}

Data assignment relation:
\{PE[i,j] \rightarrow Y[i,j]\}
\{T[i+j+k] \rightarrow Y[i,j]\}

Diagram:

- For \(t = 0\): PE[0,0] \rightarrow Y[0,0], PE[0,1] \rightarrow Y[0,1], PE[1,0] \rightarrow Y[1,0], PE[1,1] \rightarrow Y[1,1]
- For \(t = 1\): PE[0,0] \rightarrow Y[0,0], PE[0,1] \rightarrow Y[0,1], PE[1,0] \rightarrow Y[1,0], PE[1,1] \rightarrow Y[1,1]
- For \(t = 2\): PE[0,0] \rightarrow Y[0,0], PE[0,1] \rightarrow Y[0,1], PE[1,0] \rightarrow Y[1,0], PE[1,1] \rightarrow Y[1,1]
- For \(t = 3\): PE[0,0] \rightarrow Y[0,0], PE[0,1] \rightarrow Y[0,1], PE[1,0] \rightarrow Y[1,0], PE[1,1] \rightarrow Y[1,1]
PE interconnection relation

Systolic relation

\{PE[i,j] \rightarrow PE[i,j + 1]\}

\{PE[i,j] \rightarrow PE[i + 1,j]\}

- At t = 0:
  - PE[0,0] connections to PE[0,1]
  - A[0][0] connections to PE[0,1]

- At t = 1:
  - PE[0,1] connections to PE[0,2]
  - A[0][0] connections to PE[0,1]

Scratchpad: read twice
Performance model

Each relation is an integer set

- \{A[0,0] \rightarrow T[0]\}
- \{A[0,3] \rightarrow T[3]\}
- \{A[0,2] \rightarrow T[3]\}
- \{A[1,2] \rightarrow T[3]\}
- \{A[1,1] \rightarrow T[3]\}

Reuse Volume

Unique Volume

Total Volume

Data assignment relation set

tensor iteration domain

data movement across time
**ReuseVolume example**

\[
\begin{align*}
\text{set}(T[2]) & \quad \text{set}(T[3]) \quad \text{reused data} \\
\{A[0, 2] \rightarrow T[2]\} & \quad \{A[0, 3] \rightarrow T[3]\} & A[0, 2], A[1, 1] \\
\{A[0, 1] \rightarrow T[2]\} & \quad \{A[0, 2] \rightarrow T[3]\} \\
\{A[1, 1] \rightarrow T[2]\} & \quad \{A[1, 2] \rightarrow T[3]\} \\
\{A[1, 0] \rightarrow T[2]\} & \quad \{A[1, 1] \rightarrow T[3]\}
\end{align*}
\]

\[
\bigcap \quad \bigcap
\]

Check:
1. is in the same PE
2. via interconnection

\[
\text{set}(T[0]) \cap \text{set}(T[1]) + \text{set}(T[1]) \cap \text{set}(T[2]) + \ldots + \text{set}(T[n-2]) \cap \text{set}(T[n-1]) = \text{ReuseVolume}
\]
Spatial reuse and temporal reuse

Spatial reuse involves the reuse of spatial stamps in the same PE to avoid overlap.

Temporal reuse involves the reuse of multiple time-stamps in the same PE.
Use Volumes to calculate reuse and latency

average data reuse
\[ \text{ReuseFactor} = \frac{\text{TotalVolume}}{\text{UniqueVolume}} \]

Read latency
\[ \frac{\text{UniqueVolume(\text{Input tensor})}}{\text{scratchpad-bandwidth}} \]

Store latency
\[ \frac{\text{UniqueVolume(\text{Output tensor})}}{\text{scratchpad-bandwidth}} \]

Compute latency
\[ \frac{\text{sum(\text{instances})}}{\text{AVG(activated PEs)}} \]

Total latency = MAX(read, compute, store)
Use Volumes to calculate bandwidth

**Required NoC bandwidth**
SpatialReuseVolume / compute latency

**Required scratchpad bandwidth**
UniqueVolume / compute latency
Tutorial: how to use TENET

./bin/tenet -h
STEP 1: describe a statement

Statement file: conv.s

2D-convolution written in relation:

```
2 1
// 2 means two 2 input tensors, 1 means one output tensor

{S[k,c,ox,oy,rx,ry]: 0<=k<128 and 0<=c<64 and 0<=ox<112 and 0<=oy<112 and 0<=rx<3 and 0<=ry<3}
// specify the loop boundary

{S[k,c,ox,oy,rx,ry]->I[c,ox+rx,oy+ry]}
// specify the access function of input image tensor

{S[k,c,ox,oy,rx,ry]->W[k,c,rx,ry]}
// specify the access function of weight tensor

{S[k,c,ox,oy,rx,ry]->O[k,ox,oy]}
// specify the access function of output image tensor
```

**Assumption**: output is generated by multiply-and-add

\[ O[k,ox,oy] += I[c,ox+rx,oy+ry] \times W[k,c,rx,ry] \]
STEP 2: specify the PE array

PE array file: pe_array.p

8x8 systolic array:

\{PE[i,j]:0\leq i<8 \text{ and } 0\leq j<8\}
//specify the PE array size

\{PE[i,j]->PE[i+1,j]; \ PE[i,j]->PE[i,j+1]\}
// specify the systolic interconnection

128 1024 64 4
//L1 size or scratchpad size
//L2 size or DRAM size
//bandwidth(element/cycle)
//average pipeline depth, equal to the half of PE array width
STEP 3: specify the time-stamp

Mapping file: dataflow.m

map instance to space-stamp and time-stamp:

map the loop index to the PE index:

\{S[k,c,ox,oy,rx,ry] \rightarrow PE[k%8,c%8]\}

The systolic access pattern requires the inner-most time-stamp to be:

\{S[k,c,ox,oy,rx,ry] \rightarrow T[k%8 + c%8 + ox]\}

Map the outer loop index to the time-stamp:

\{S[k,c,ox,oy,rx,ry] \rightarrow T[\text{floor}(k/8),\text{floor}(c/8),oy,k%8 + c%8 + ox]\}

So the complete dataflow is:

\{S[k,c,ox,oy,rx,ry]\rightarrow PE[k%8,c%8]\} // space-stamp: instance to PE

\{S[k,c,ox,oy,rx,ry]\rightarrow T[\text{floor}(k/8),\text{floor}(c/8),oy,k%8 + c%8 + ox]\} //time-stamp: instance to cycle number
Tutorial: model a dataflow (4/4)

STEP 4: run TENET

tenet –m dataflow.m –p pe_array.p –s conv.s –o test.csv --all

output to a csv file
0.1s to evaluate a dataflow on a 2-core 2.50GHz Intel Core i5-7200U CPU

**Complexity of interconnection topology**

40min to finish the DSE of 2D-CONV

**Number of loops**
Metrics evaluation (2D-CONV)

Different dataflows

{S[k,c,ox,oy,rx,ry]->PE[k%8,c%8]}
// space-stamp: instance to PE

{S[k,c,ox,oy,rx,ry]->T[floor(k/8),floor(c/8),rx,ry,oy,ox]}
// time-stamp: instance to cycle number

only keep inner-most two time dimensions
### Notate dataflows

<table>
<thead>
<tr>
<th>Tensor kernel</th>
<th>Dataflow</th>
<th>Relation-centric notation</th>
<th>Data-centric notation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(KJ-P</td>
<td>K,JK-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(IK-P</td>
<td>K,IK-T)</td>
<td>✓</td>
</tr>
<tr>
<td><strong>2D-CONV</strong></td>
<td>(KOX-P</td>
<td>OY,KOXC-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(KC-P</td>
<td>OY,KCOX-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(K-P</td>
<td>OX,OY-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(RYOY-P</td>
<td>OY,OX-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(I-P</td>
<td>I,J-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(IJ-P</td>
<td>I,J-T)</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Jacobi-2D</strong></td>
<td>(I-P</td>
<td>I,J-T)</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>(IJ-P</td>
<td>I,J-T)</td>
<td>✓</td>
</tr>
<tr>
<td><strong>MMc</strong></td>
<td>(IJ-P</td>
<td>J,IJL-T)</td>
<td>✓</td>
</tr>
<tr>
<td>(Attention mechanism)</td>
<td>(KJ-P</td>
<td>J,KJL-T)</td>
<td>✓</td>
</tr>
</tbody>
</table>

**More expressive, more optimization opportunities**

**Support more tensor kernels**
Bandwidth analysis under different interconnect

IBW: interconnection bandwidth. SBW: scratchpad bandwidth

Increasing connections does not necessarily reduce scratchpad bandwidth requirement. Interconnection network needs to take the data movement patterns into account.
Tutorial: model a network (1/2)

STEP 1: set the test file

MobileNet_config_dir

layer_1_config

layer_2_config

layer_3_config

dataflow.m
pe_array.p
statement_layer1.s

dataflow.m
pe_array.p
statement_layer2.s

dataflow.m
pe_array.p
statement_layer5.s
Tutorial: model a dataflow (2/2)

**STEP 2: run TENET**

```bash
$ tenet -e ./network_example/MobileNet/config -d ./network_example -o test.csv
```

TENET will analyze each layer in sequence. If no `--all`, TENET only shows partial results.
1. TENET supports output reuse analysis.
2. TENET supports multi-dimensional time-stamps
3. TENET supports quasi-affine transformation
Summary

- A framework analyze tensor dataflow: **TENET**
- Relation-centric notation
  - More expressive
- A performance model
  - More precise
- Open source: [https://github.com/pku-liang/TENET](https://github.com/pku-liang/TENET)
- Document: [https://tenet-docs.readthedocs.io/](https://tenet-docs.readthedocs.io/)