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

Liqiang Lu\textsuperscript{1}, Naiqing Guan\textsuperscript{2}, Yuyue Wang\textsuperscript{1}, Liancheng Jia\textsuperscript{1}, Zizhang Luo\textsuperscript{1}, Jieming Yin\textsuperscript{3}, Jason Cong\textsuperscript{4}, Yun Liang\textsuperscript{1}

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

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

Tensor computation

- deep learning: CONV, GEMV, GEMM, GEMMc
- image processing: CONV, Stencil, Jacobi-2D
- recommendation system: GEMM, MTTKRP, TTMc
Tensor computation

GEMM

MTTKRP

2D-CONV

TTMc

Stencil

Various tensor size & different data reduction

Regular computation & huge computation size
Tensor-specific accelerators

Google TPU

Cambricon MLU270

Huawei Ascend

Spatial architectures

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

Parallel computation

Google TPU
systolic dataflow

Cambricon MLU270
multicast dataflow

Huawei Ascend
3D-dataflow

Data movement

Matrix Func.
unit

16^3 Cube
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

instance assignment

S[0,0,0] S[0,0,1] S[0,1,1]

cycles
evaluation sequence

dataflow

data reuse
throughput

data access

bandwidth

latency
Existing notations

Compute-centric notation:

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

```cpp
do in pipeline
for (j = 0; j < 3; j++)
    do in parallel:
        for (i = 0; i < 4; i++)
            S: Y[i] += A[i+j]*B[j];
```

Data-centric notation:

```cpp
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

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

**Data-centric notation reference:**
Kwon, Hyoukjun, et al. "Understanding reuse, performance, and hardware cost of dnn dataflow: A data-centric approach." MICRO, 2019.
Design space of existing notations

Dataflow example

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

$T[i+j] \to A[i,j]$
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

Dataflow relation set

{S[0,0,0] → T[0]}
{S[0,0,1] → T[1]}
{S[0,1,0] → T[1]}
{S[1,0,0] → T[1]}
{S[0,1,1] → T[1]}
{S[1,0,1] → T[1]}
{S[2,2,2] → T[8]}

Dataflow relation set

iteration numbers

3 × 3 × 3
3 × 3

= 3

parallel factor

find max T {S[2,2,2] → T[8]}

investigate each point
in the dataflow set

more precise
Relation-centric notation overview

Dataflow relation

Data assignment relation

Interconnect relation

Precisely model the tensor dataflow on spatial architectures
Example: matrix multiplication

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

Dataflow 1:
Systolic array
output stationary

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

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

Dataflow 2:
Systolic array
input stationary

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

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

affine transformation

\[
\text{loop nest: }
\]
\[
\text{for } (i = 0; i < 3; i++)
\]
\[
\text{for } (j = 0; j < 3; j++)
\]
\[
\text{for } (k = 0; k < 3; k++)
\]
\[
\text{instance: } S[i,j,k]:
\]
\[
Y[i,j] += A[i,k] \times B[k,j];
\]
Step by step example

instance operation

\[ S[i,j,k] \rightarrow A[i,k]B[k,j] \]

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

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:

- t = 0
  - PE[0,0]
  - PE[0,1]
  - Y[0][0]
  - Y[0][1]
  - PE[1,0]
  - PE[1,1]
  - Y[1][0]
  - Y[1][1]

- t = 1
  - PE[0,0]
  - PE[0,1]
  - Y[0][0]
  - Y[1][0]
  - PE[1,0]
  - PE[1,1]
  - Y[0][1]
  - Y[1][1]

- t = 2
  - PE[0,0]
  - PE[0,1]
  - Y[0][0]
  - Y[0][1]
  - PE[1,0]
  - PE[1,1]
  - Y[1][0]
  - Y[1][1]

- t = 3
  - PE[0,0]
  - PE[0,1]
  - Y[0][0]
  - Y[0][1]
  - PE[1,0]
  - PE[1,1]
  - Y[1][0]
  - Y[1][1]
**PE interconnection relation**

PE\[i\]  
\[\rightarrow\]  
PE\[i'\]

**Systolic relation**

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

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

---

$t = 0$

**reuse**

| PE[0,0] |
|----------|
| A[0][0]  |

$t = 1$

**read twice**

| PE[0,1] |
|----------|
| A[0][0]  |

**scratchpad**
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
ReuseVolume example

\[ \text{set}(T[2]) \cap \text{set}(T[3]) = \text{reused data} \]

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

**Check:**
1. is in the same PE
2. via interconnection
Spatial reuse and temporal reuse

Spatial reuse volume used in multiple space-stamps

Temporal reuse volume used in multiple time-stamps in the same PE

at the same PE to avoid overlap
Use Volumes to calculate reuse and latency

average data reuse
ReuseFactor = TotalVolume / UniqueVolume

Read latency
UniqueVolume(Input tensor) / scratchpad-bandwidth

Store latency
UniqueVolume(Output tensor) / scratchpad-bandwidth

Compute latency
sum(instances) / 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 \text{ and } 0<=c<64 \text{ and } 0<=ox<112 \text{ and } 0<=oy<112 \text{ and } 0<=rx<3 \text{ and } 0<=ry<3\}
// specify the loop boundary

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

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

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

Assumption: output is generated by multiply-and-add

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

PE array file: pe_array.p

8x8 systolic array:

{PE[i,j]:0<=i<8 and 0<=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 \text{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
**Runtime results**

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

abbreviated as (KC-P | Oy,Ox-T)

long latency

- temporal reuse data
- spatial reuse data
- max PE utilization
- avg PE utilization
- latency

2D-CONV

Latency (10^7)

Norm. reuse & PE utilization
## Notate dataflows

| Tensor kernel | Dataflow | Relation-centric notation | Data-centric notation |
|---------------|----------|---------------------------|-----------------------|
| GEMM          | (KJ-P | K,IJK-T) | √ | × |
|               | (IK-P | K,IJK-T) | √ | × |
| 2D-CONV       | (KOX-P | OY,KOXC-T) | √ | × |
|               | (KC-P | OY,KCOX-T) | √ | × |
|               | (K-P | OX,OY-T) | √ | √ |
|               | (RYOY-P | OY,OX-T) | √ | ✓ |
| MTTKRP        | (IJ-P | J,IJL-T) | √ | × |
|               | (KJ-P | J,KJL-T) | √ | × |
|               | (KL-P | L,KLJ-T) | √ | × |
| Jacobi-2D     | (I-P | I,J-T) | √ | × |
|               | (IJ-P | I,J-T) | √ | × |
| MMc           | (IJ-P | J,IJL-T) | √ | × |
| (Attention mechanism) | (KJ-P | J,KJL-T) | √ | × |

### More expressive, more optimization opportunities

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

8x8 1D-systolic array (vertical):
\{PE[i,j]: 0 <= i < 8 and 0 <= j < 8\}
\{PE[i,j] -> PE[i,j+1]\}

8x8 2D-systolic array:
\{PE[i,j]: 0 <= i < 8 and 0 <= j < 8\}
\{PE[i,j] -> PE[i+1,j]; PE[i,j] -> PE[i,j+1]\}

8x8 2D-systolic array:
\{PE[i,j]: 0 <= i < 8 and 0 <= j < 8\}
\{PE[i,j] -> PE[i+1,j]; PE[i,j] -> PE[i,j+1];
PE[i,j] -> PE[i-1,j-1]; PE[i,j] -> PE[i-1,j+1];
PE[i,j] -> PE[i+1,j-1]; PE[i,j] -> PE[i+1,j+1];}\n
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
    - dataflow.m
    - pe_array.p
    - statement_layer1.s
  - layer_2_config
    - dataflow.m
    - pe_array.p
    - statement_layer2.s
  - layer_3_config
    - dataflow.m
    - pe_array.p
    - statement_layer5.s
STEP 2: run TENET

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

test.csv
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/)