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

Liqiang Lu[1], Naiqing Guan[2], Yuyue Wang[1], Liancheng Jia[1],
Zizhang Luo[1], Jieming Yin[3], Jason Cong[4], **Yun Liang**[1]

1 CECA, School of EECS, Peking University

2 Computer Science Department, University of Toronto

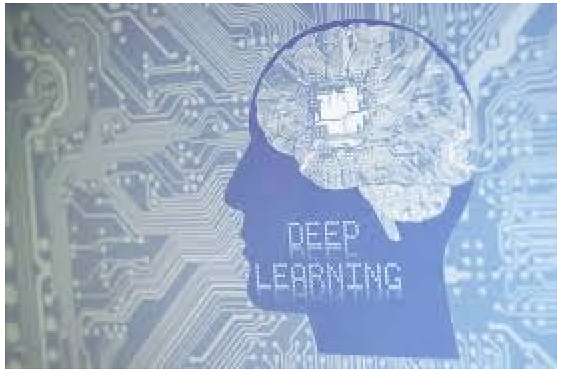
3 Electrical and Computer Engineering, Lehigh University

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

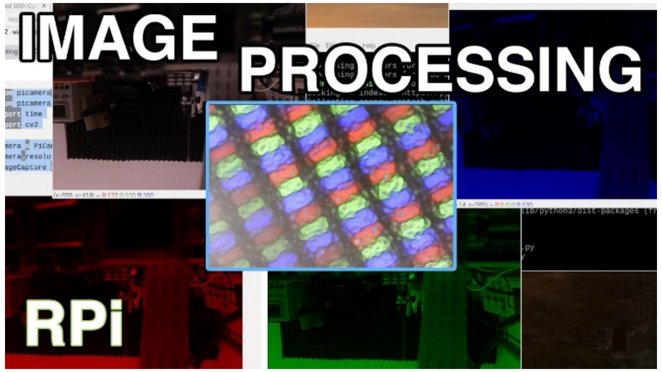
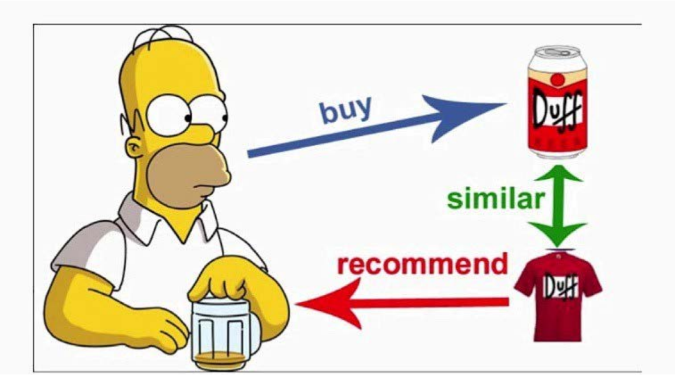


image processing



recommendation system

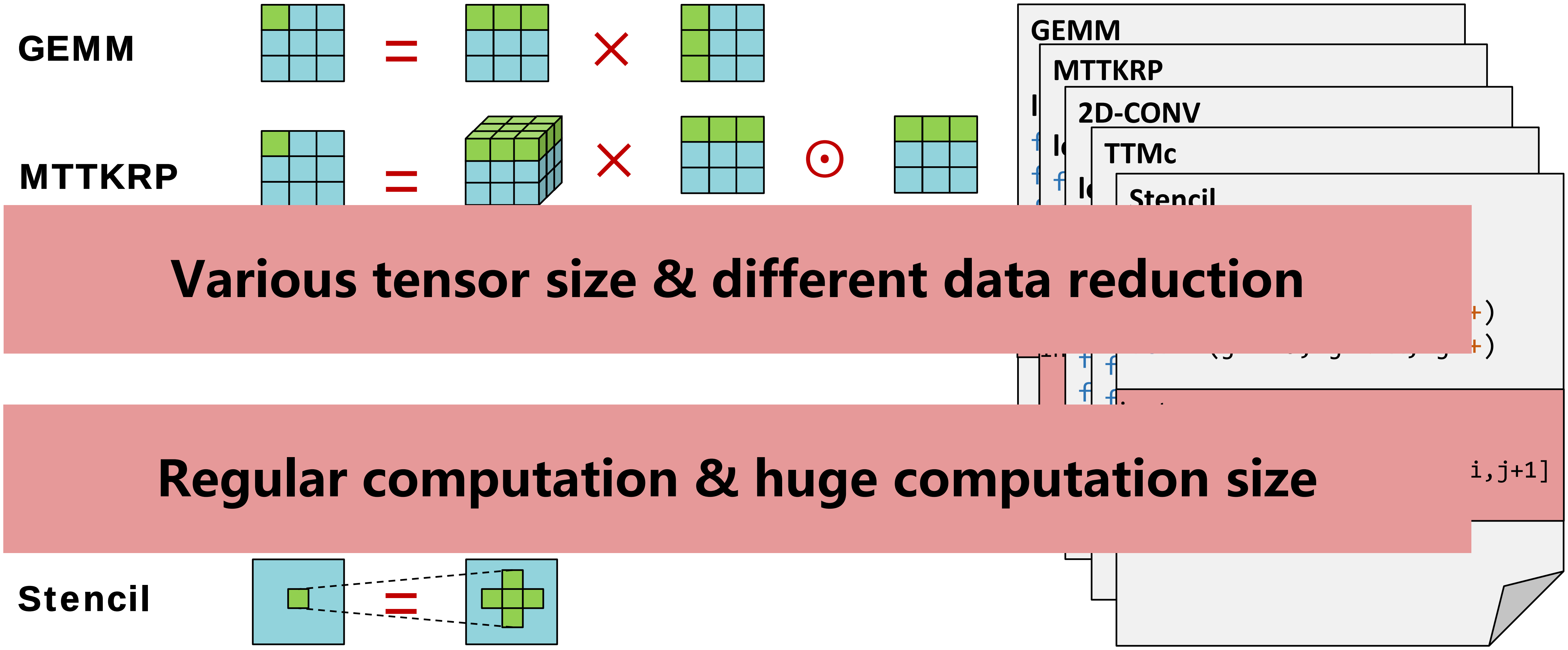
CONV, GEMV, GEMM, GEMMc

CONV, Stencil, Jacobi-2D

GEMM, MTTKRP, TTMc

Tensor computation

Tensor computation

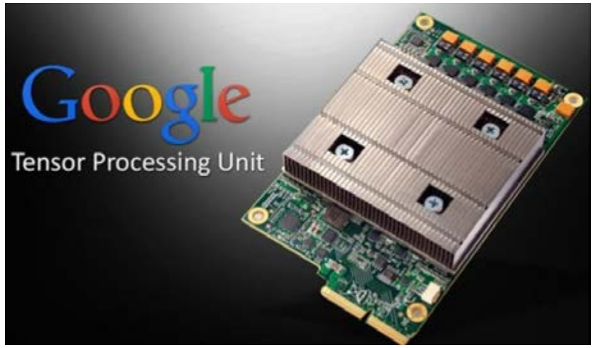


Various tensor size & different data reduction

Regular computation & huge computation size

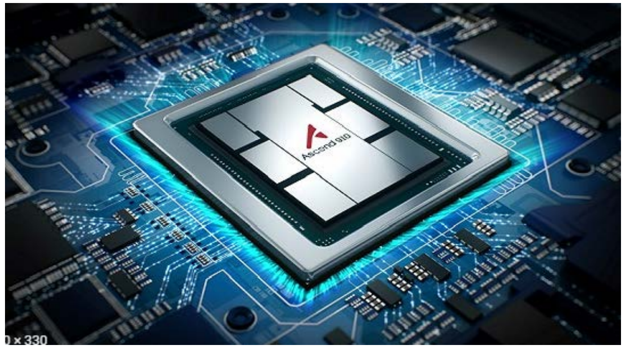
Tensor-specific accelerators

Google TPU

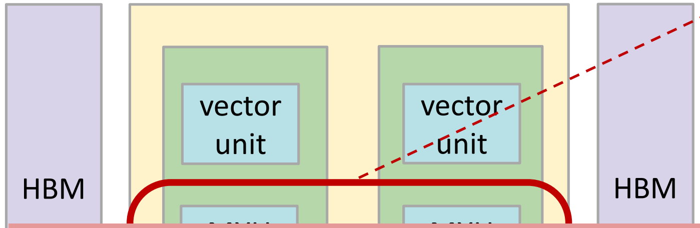


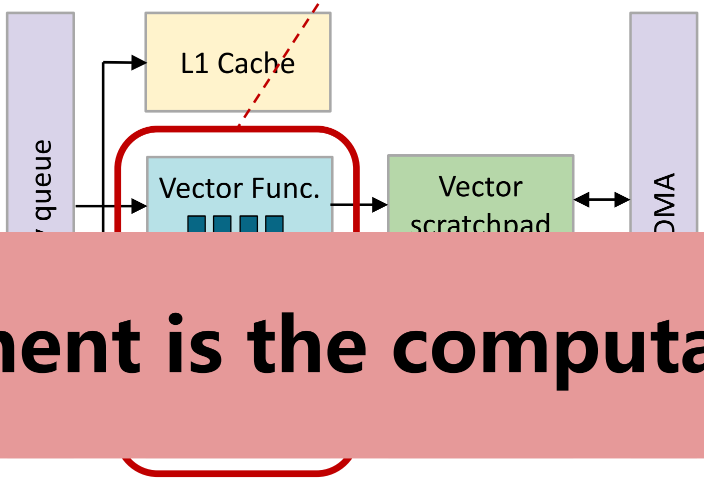
Cambricon MLU270

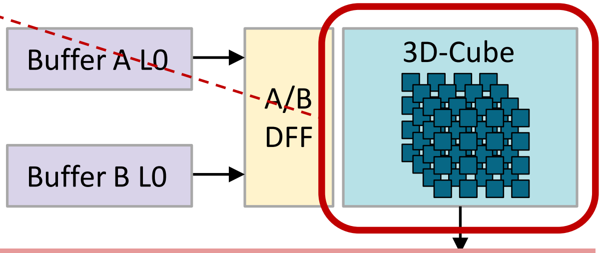
Huawei Ascend



Spatial architectures







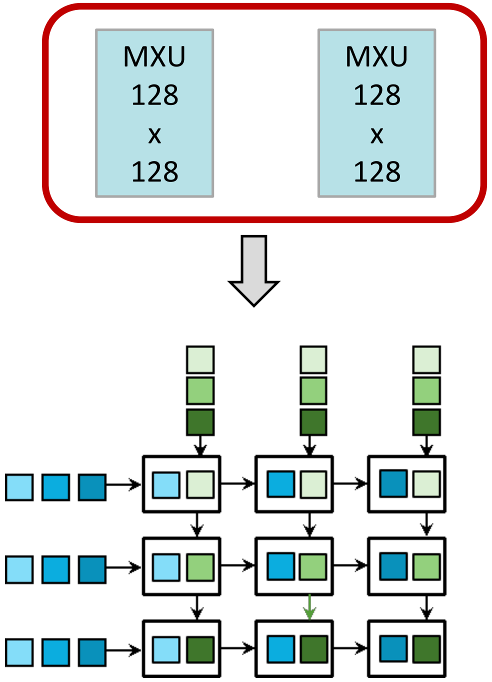
The key component is the computation dataflow

What is tensor computation dataflow ?

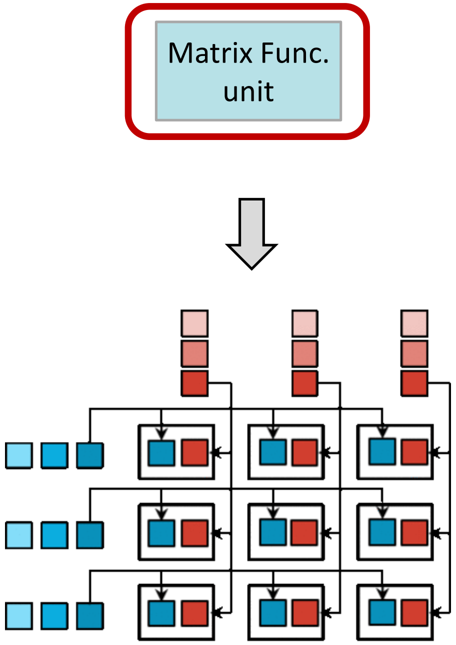
Parallel computation

Data movement

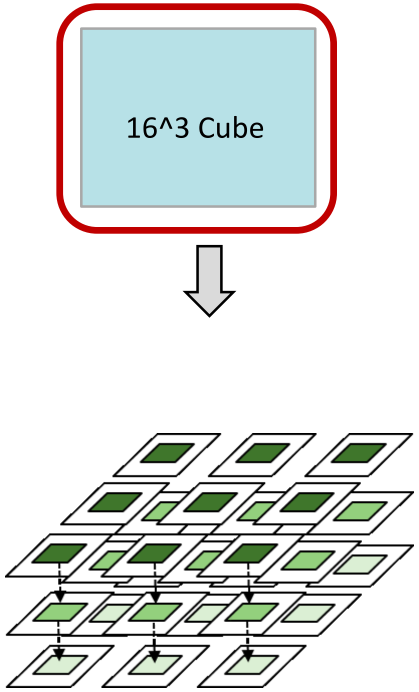
Google TPU
systolic dataflow



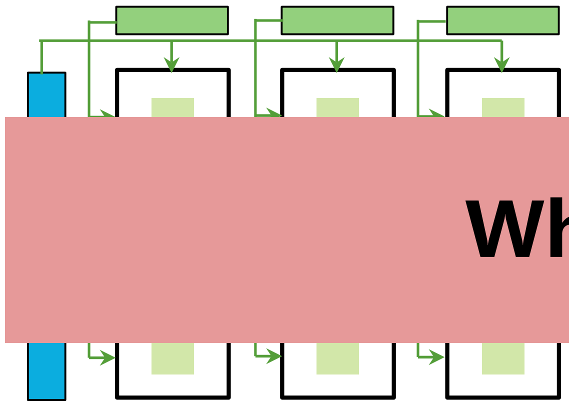
Cambricon MLU270
multicast dataflow



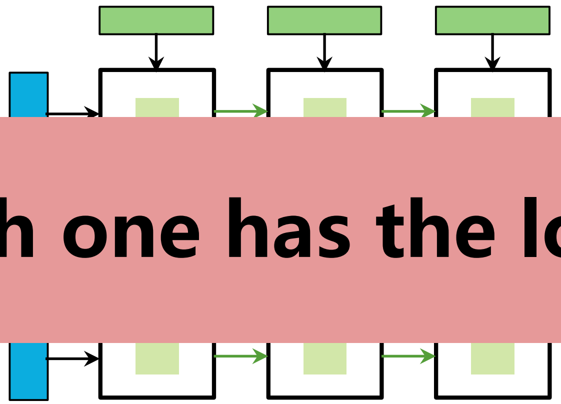
Huawei Ascend
3D-dataflow



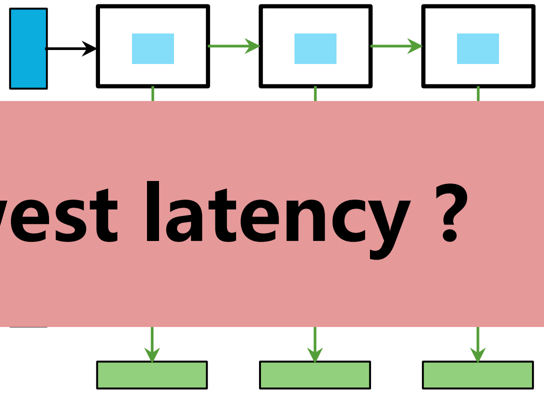
Various tensor dataflows



Multicast

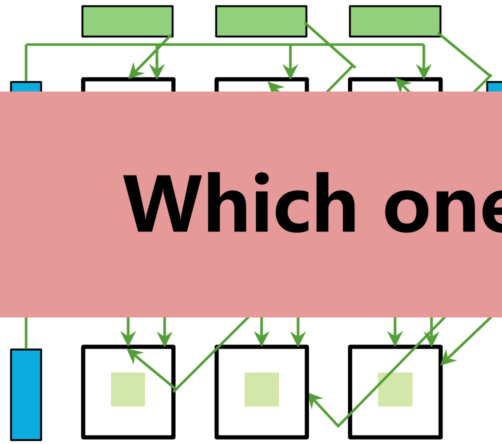


Output stationary systolic



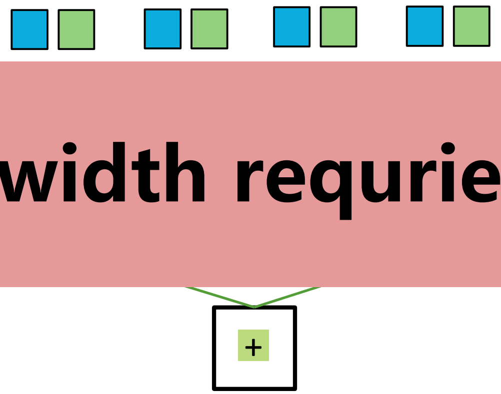
Input stationary systolic

Which one has the lowest latency ?



Eyeriss Row Stationary

Multicast+systolic



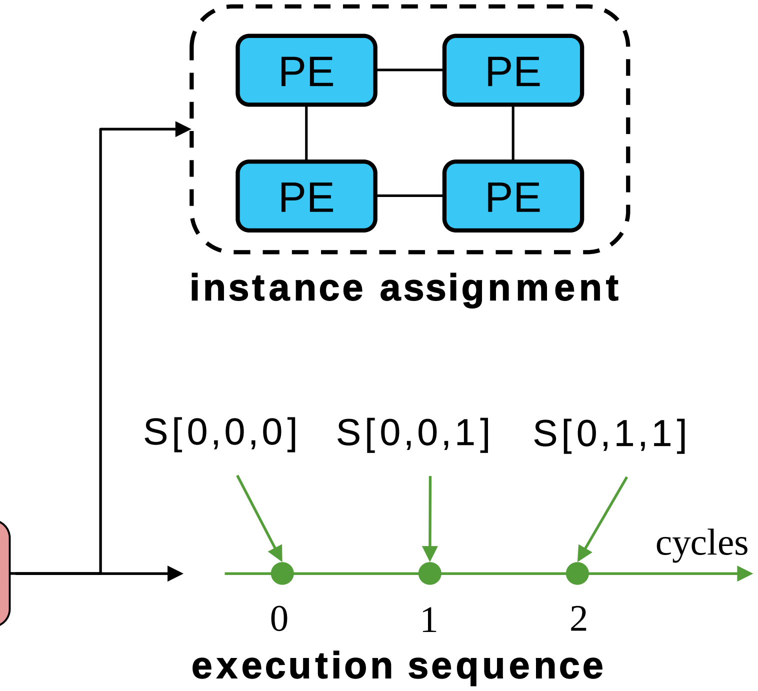
Reduction Tree

Which one has the lowest bandwidth requriement ?

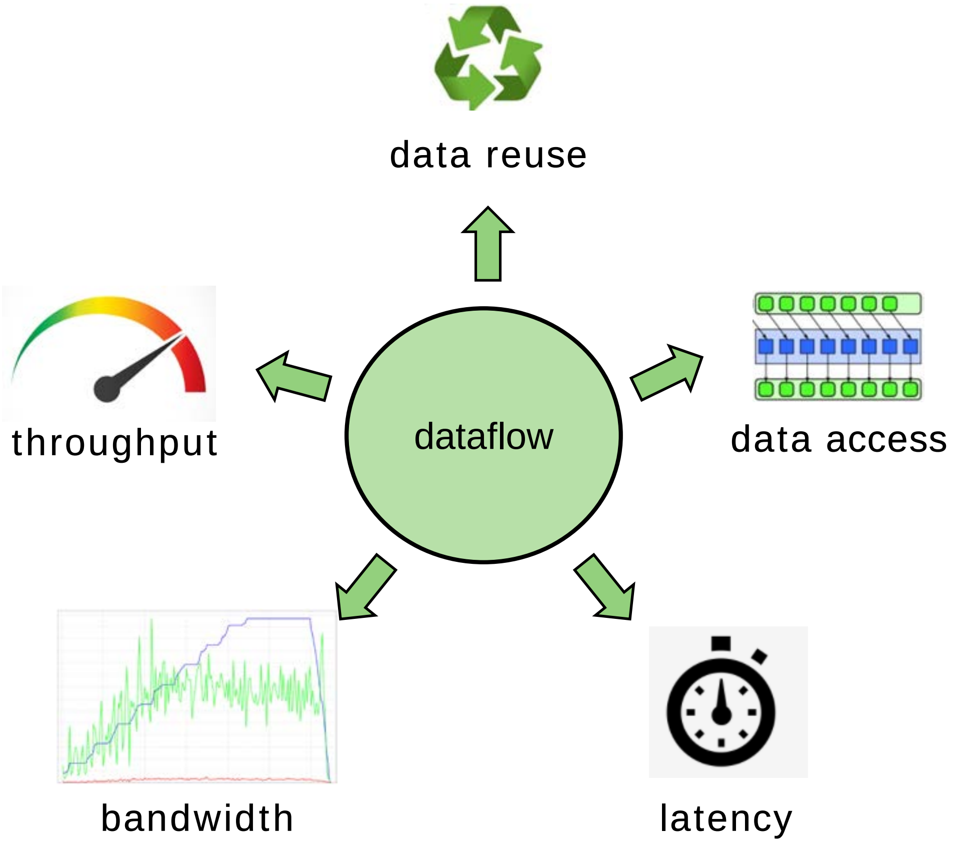
We need a framework to analyze the dataflow

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



Existing notations

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

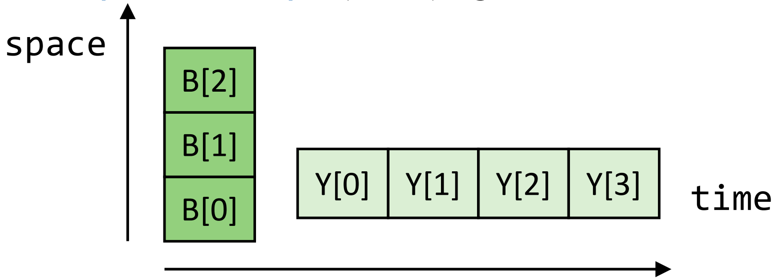
Compute-centric notation:

```
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];
```

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

Data-centric notation:

```
spatial map (1,1) i
temporal map (1,1) j
```



less expressive

limited dataflow design space

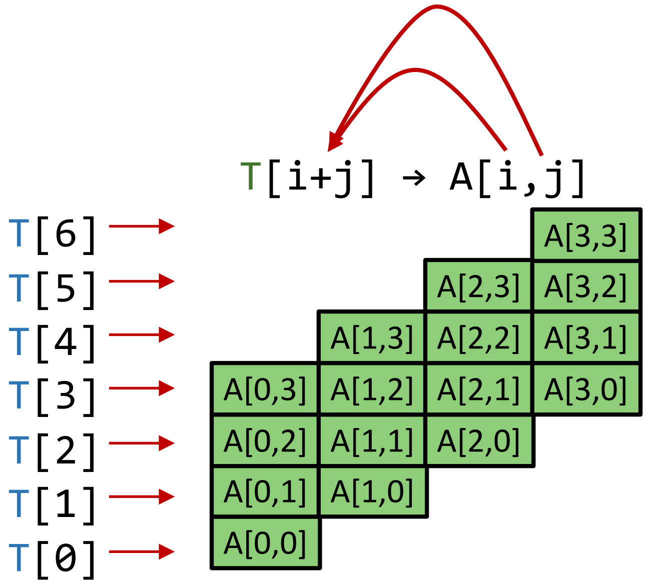
less opportunities for optimization

Compute-centric notation reference:
Yang, Xuan, et al. "Interstellar: Using Halide's Scheduling Language to Analyze DNN Accelerators." ASPLOS, 2020
Data-centric notation reference:
Kwon, Hyoukjun, et al. "Understanding reuse, performance, and hardware cost of dnn dataflow: A data-centric approach." MICRO, 2019.

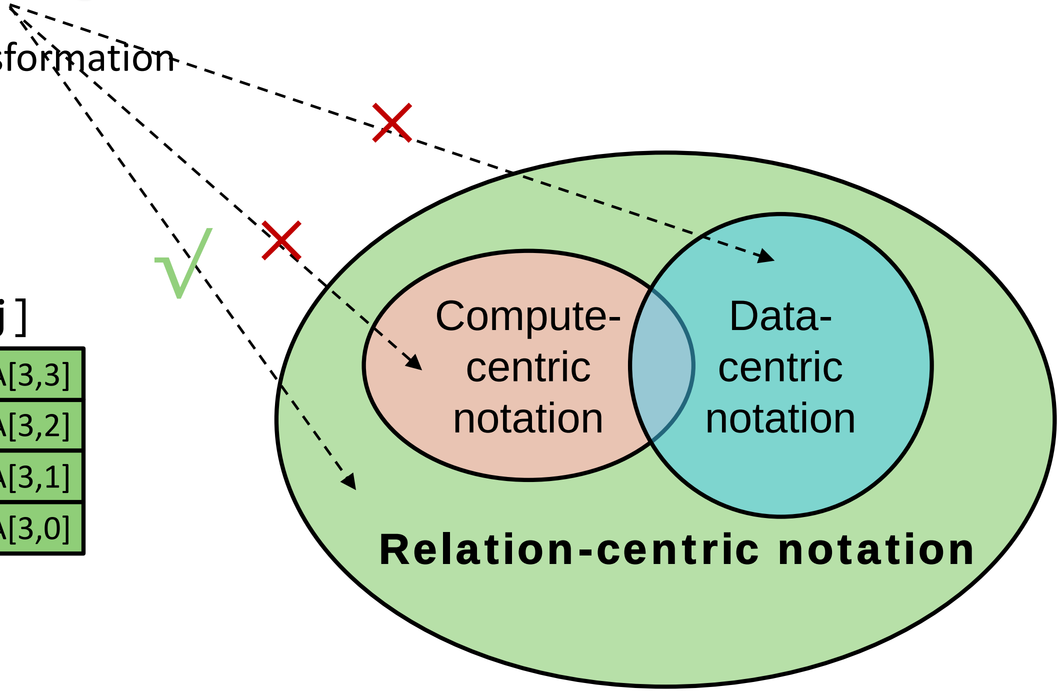
Design space of existing notations

Dataflow example

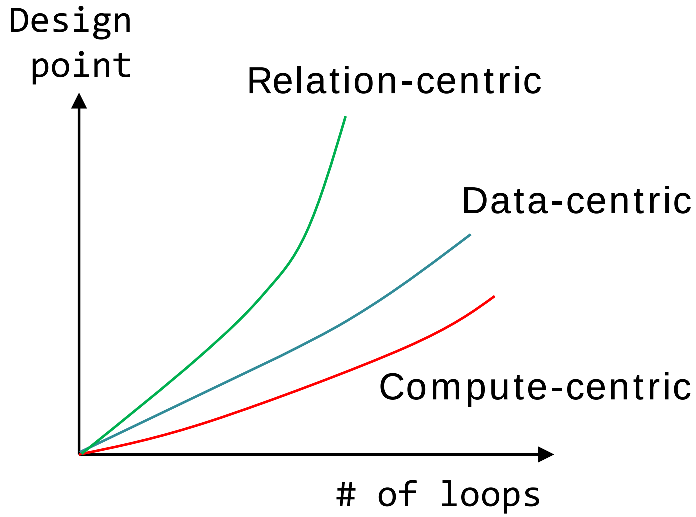
The time-stamp is an affine transformation of multiple loop dimensions



skewed data access



More expressive



A complete design space

Performance model of existing notations

```
loop nest:
for  (k = 0; k < 3; k++)
do in paralle:
  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];
```

latency

iteration numbers

$$\frac{3 \times 3 \times 3}{3 \times 3} = 3$$

parallel factor

Simple polynomials!

```
{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[2,2,2] → T[8]}
```

Dataflow relation set

latency

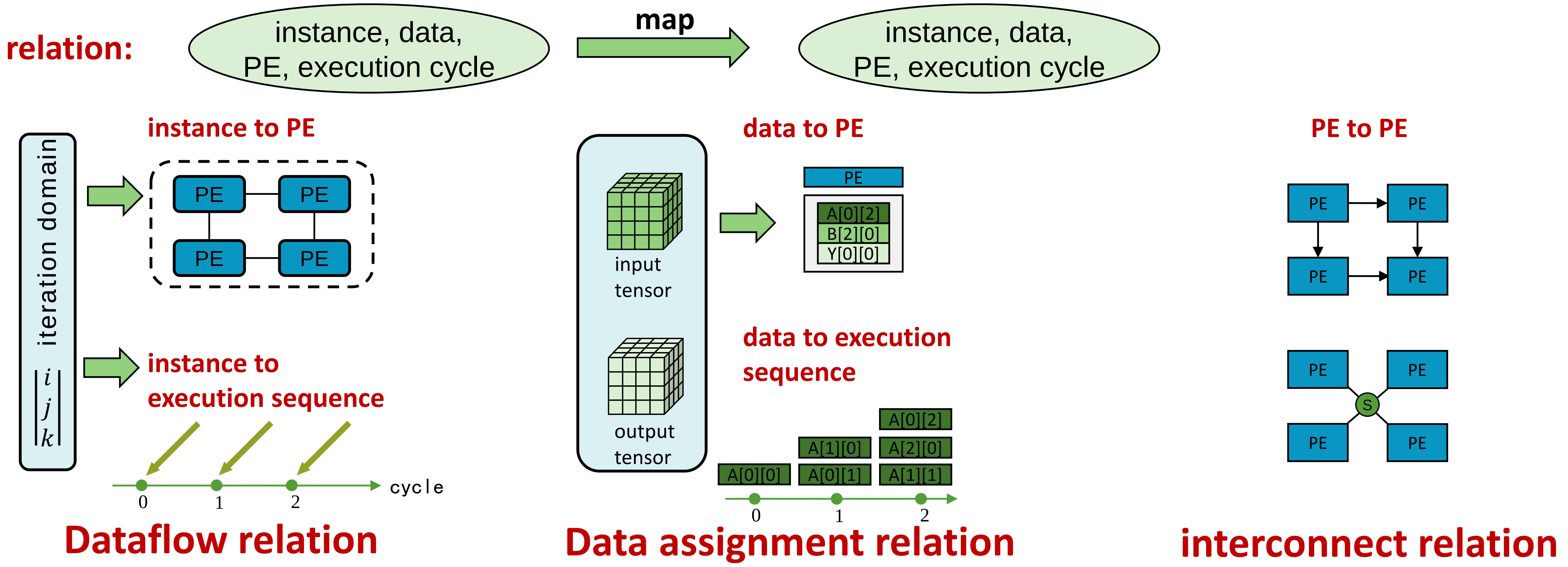
find max T {S[2,2,2] → T[8]}

Integer set operators

investigate each point in the dataflow set

more precise

Relation-centric notation overview



Precisely model the tensor dataflow on spatial architectures

Dataflow relation

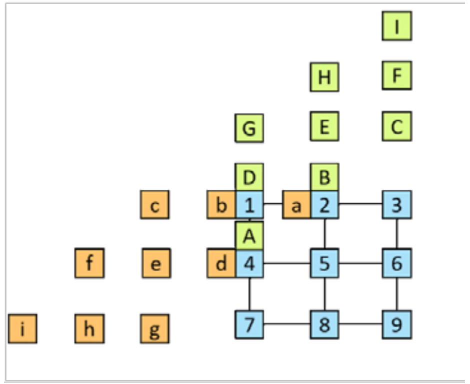
Example:
matrix multiplication

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

input output

```
loop nest:
for  (i = 0; i < 3; i++)
  for  (j = 0; j < 3; j++)
    for  (k = 0; k < 3; k++)
    instance: S[i,j,k]:
    Y[i,j] += A[i,k] * B[k,j];
```

Dataflow 1:
Systolic array
output stationary



instance to PE

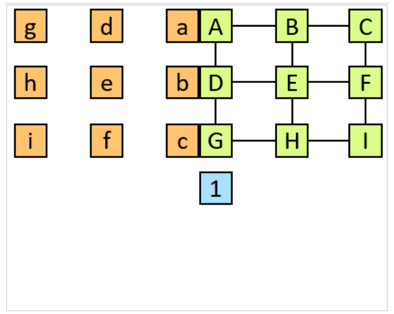
{S[i,j,k] → PE[i,j]}

instance to cycle number

{S[i,j,k] → T[i+j+k]}

affine transformation

Dataflow 2:
Systolic array
input stationary



instance to PE

{S[i,j,k] → PE[j,k]}

instance to cycle number

{S[i,j,k] → T[i+k]}

affine transformation

Step by step example

instance	operation
S[i,j,k]	A[i,k]*B[k,j]

```
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];
```

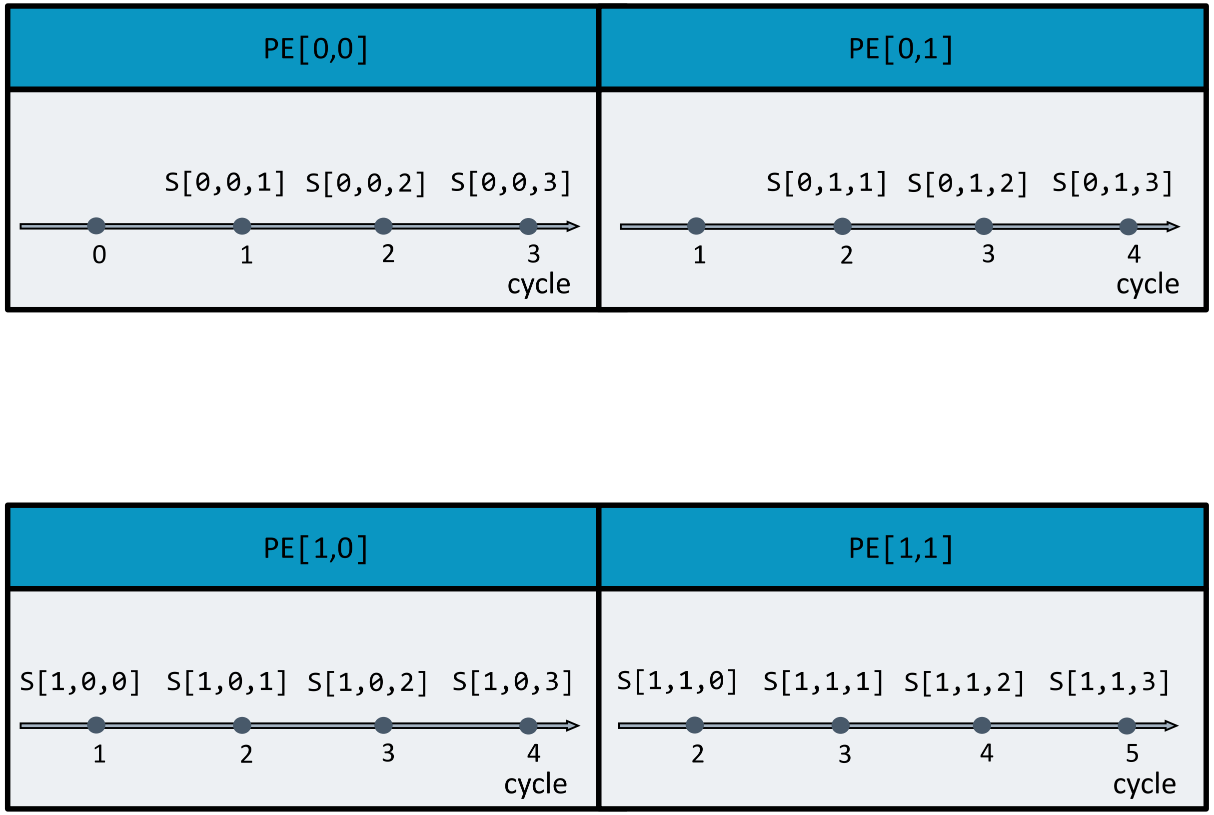
Dataflow relation

{S[i,j,k] → PE[i,j]}

obtain PE id

{S[i,j,k] → T[i+j+k]}

obtain execution cycle



Data assignment relation

Tensor kernel tells:

tensor Y to instance

{Y[i,j] → S[i,j,k]}

tensor A/B to instance

{A[i,k] → S[i,j,k]}

{B[k,j] → S[i,j,k]}

\+

Dataflow relation tells:

instance to PE

{S[i,j,k] → PE[i,j]}

instance to cycle number

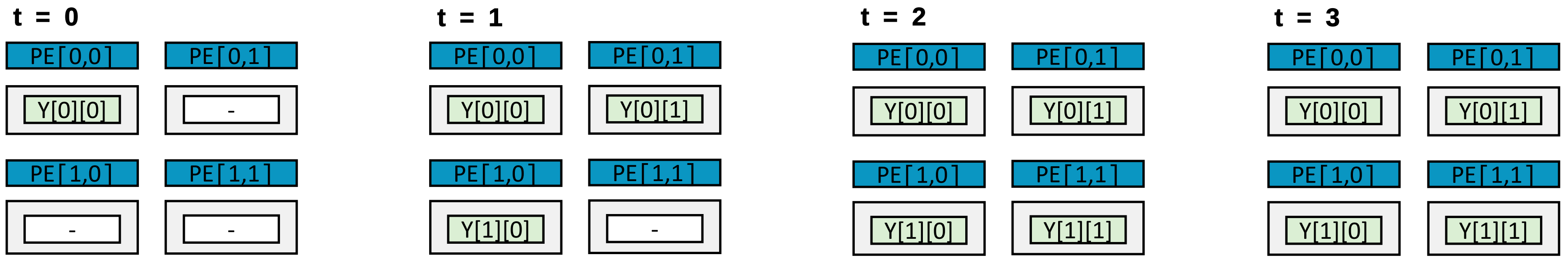
{S[i,j,k] → T[i+j+k]}

=

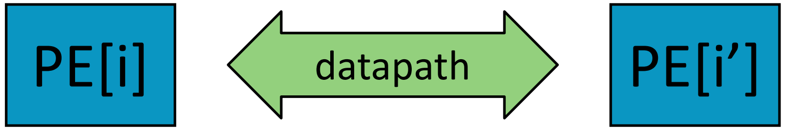
Data assignment relation:

{PE[i,j] → Y[i,j]}

{T[i+j+k] → Y[i,j]}



PE interconnection relation



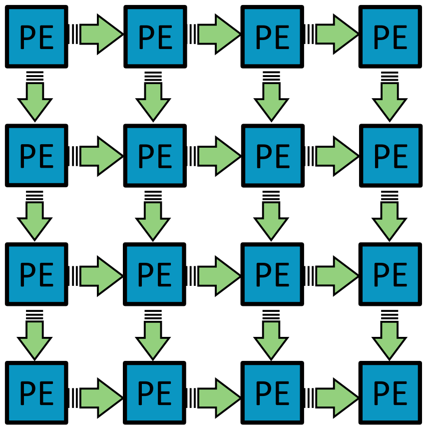
relation

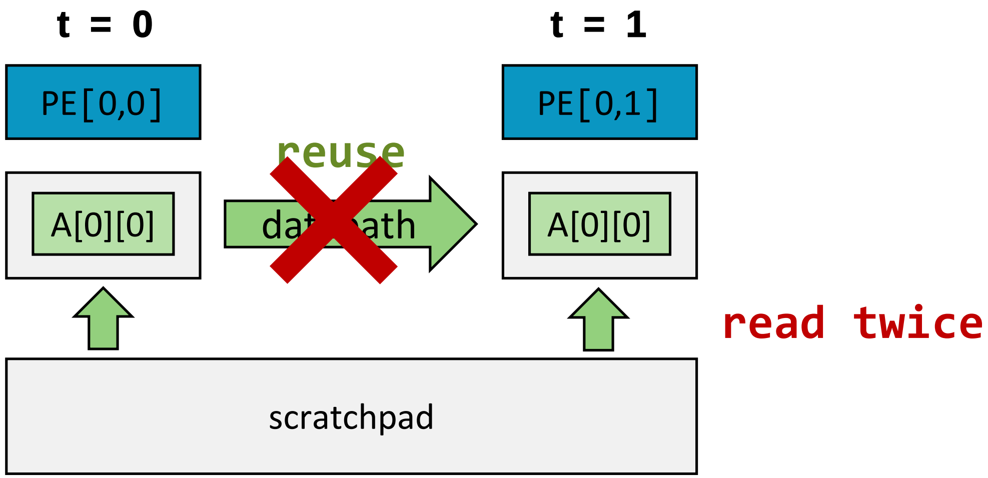
$\{\mathrm{PE}[i] \rightarrow \mathrm{PE}[i']\}$

Systolic relation

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

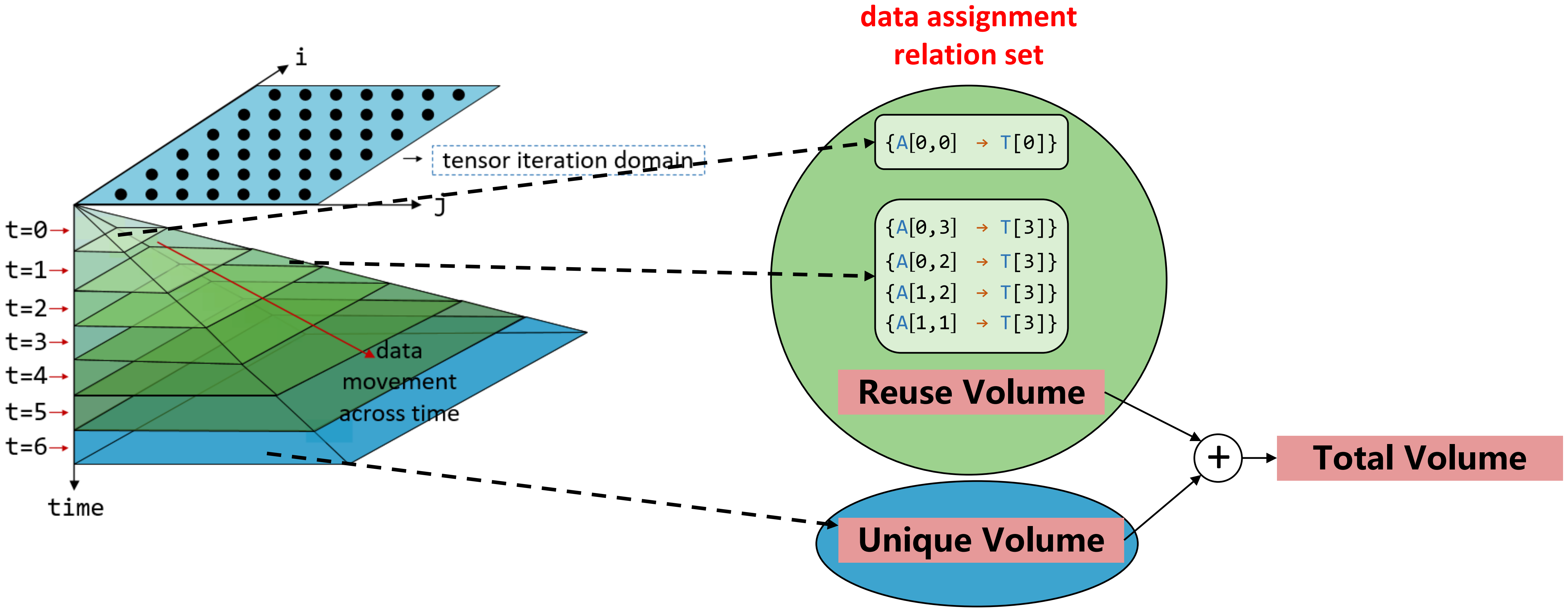
$\{\mathrm{PE}[i, j] \rightarrow \mathrm{PE}[i+1, j]\}$





Performance model

Each relation is an integer set



ReuseVolume example

set(T[2])		set(T[3])		reused data
{A[0,2] → T[2]} {A[0,1] → T[2]} {A[1,1] → T[2]} {A[1,0] → T[2]}	∩	{A[0,3] → T[3]} {A[0,2] → T[3]} {A[1,2] → T[3]} {A[1,1] → T[3]}	=	A[0,2] A[1,1]

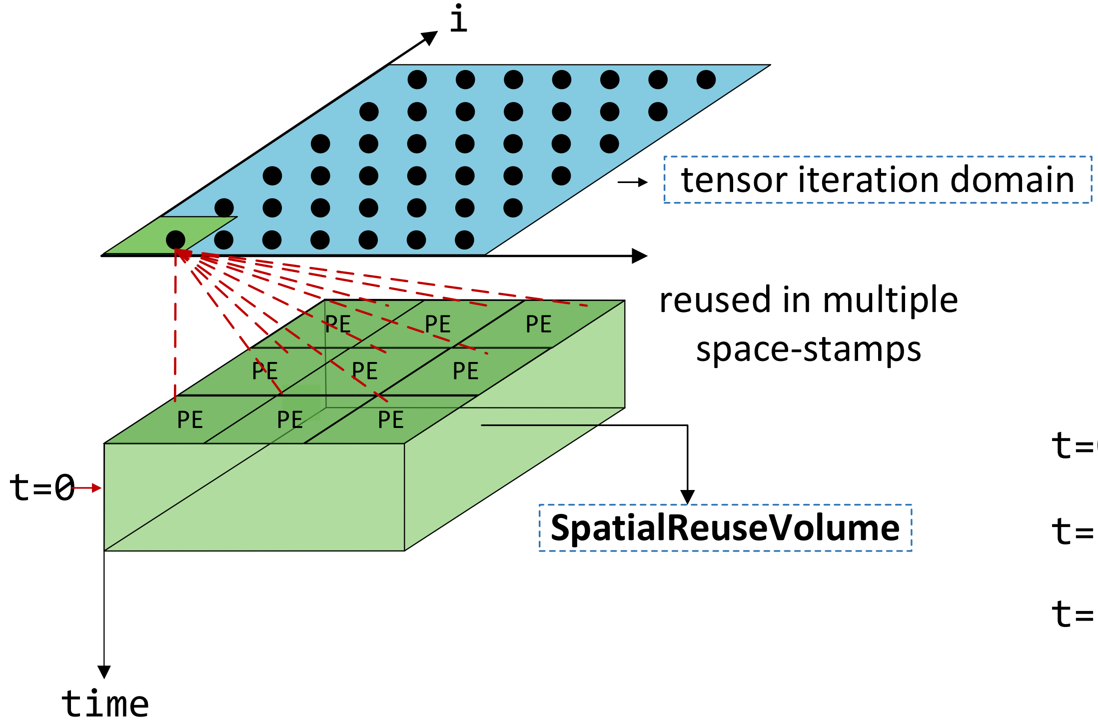
Check:

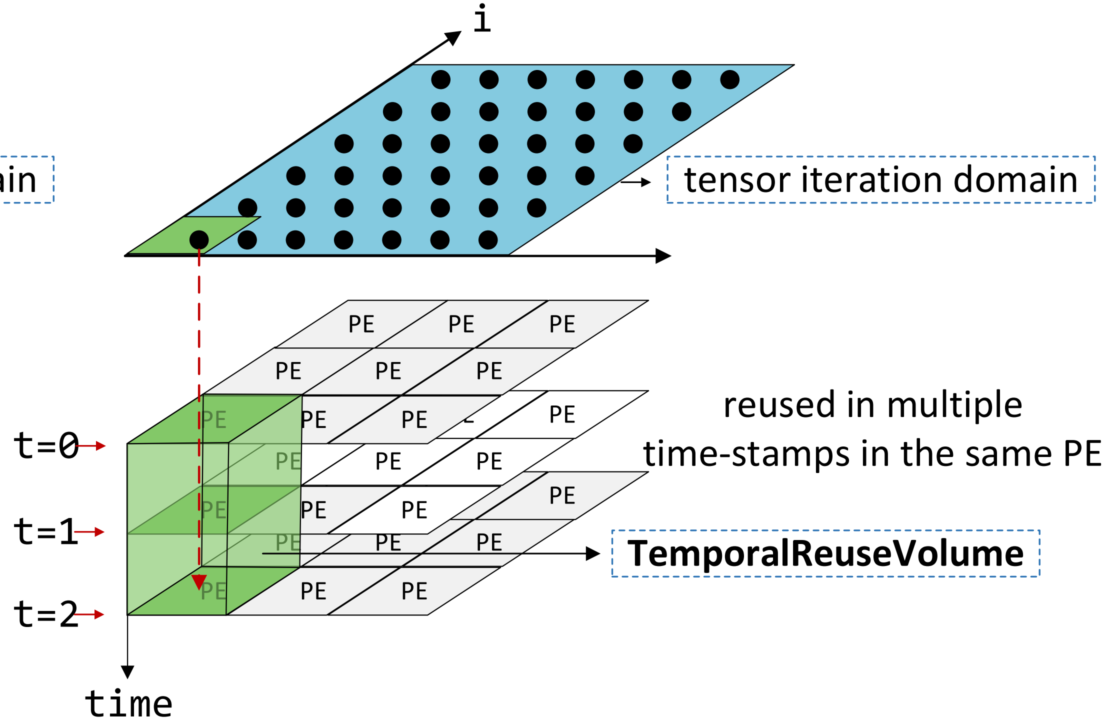
1. is in the same PE
2. via interconnection

set(T[0]) ∩ set(T[1]) + set(T[1]) ∩ set(T[2]) + ··· set(T[n-2]) ∩ set(T[n-1])

= **ReuseVolume**

Spatial reuse and temporal reuse





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

```
→  TENET git:(micro_tutorial) ✗ ./bin/tenet -h
Usage: tenet [options]
Options:
  -h, --help:                 Display this information.
  -e, --experiment <arg>:     Path of the expriment file or a directory.
                              If <arg> is a file, run the experiment file.
                              If <arg> is a directory, run all
                              experiment_? files inside of the directory.
                              When this argument is specified, -p, -m and
                              -s are ignored.
  -d, --experiment_dir <dir>: Base directory for files in the experiment.
  -p, --pe_array <file>:      Path of the pe array file.
  -m, --mapping <file>:       Path of the mapping file.
  -s, --statement <file>:     Path of the statement file.
  -o, --output <file>:        Path of the output CSV file.
  --all:                      Print all infomation.
Contact semiwaker@pku.edu.cn for bugs
```

Tutorial: model a dataflow (1/4)

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] * W[k,c,rx,ry]

Tutorial: model a dataflow (2/4)

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
```

Tutorial: model a dataflow (3/4)

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] → PE[k%8,c%8]}

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

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

Map the outer loop index to the time-stamp:

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

So the complete dataflow is:

```
{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),oy,k%8 + c%8 + ox]}
//time-stamp: instance to cycle number
```

Tutorial: model a dataflow (4/4)

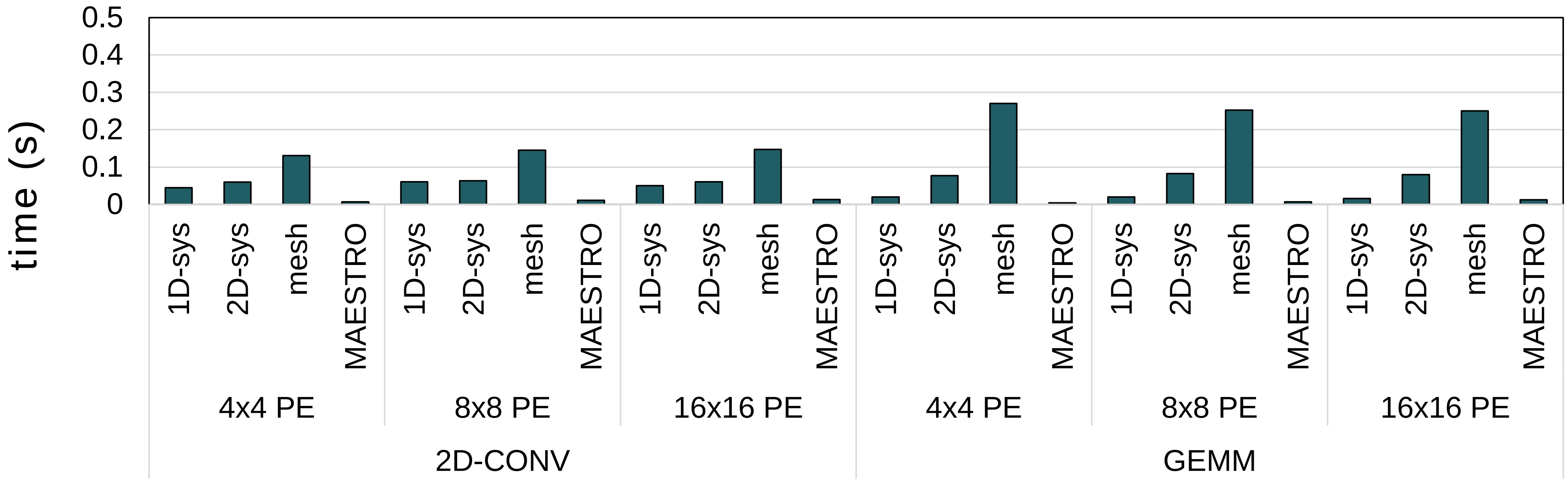
STEP 4: run TENET

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

output to a csv file

```
Delay: In: 911601; Out: 640001; Com: 914285
Active PE Num: 70; Average: 66.67
Input Tensor: I
 Reuse Factor: 17.75
 temporal: 0.943650
 spatial: 0.000000
 TotalVolume: 64000000
 UniqueVolume: 3606400
Input Tensor: W
 Reuse Factor: 1600.00
 temporal: 0.991250
 spatial: 0.925625
 TotalVolume: 64000000
 UniqueVolume: 40000
Output Tensor: O
 Reuse Factor: 25.00
 temporal: 0.800000
 spatial: 0.160000
 TotalVolume: 64000000
 UniqueVolume: 2560000
```

Runtime results



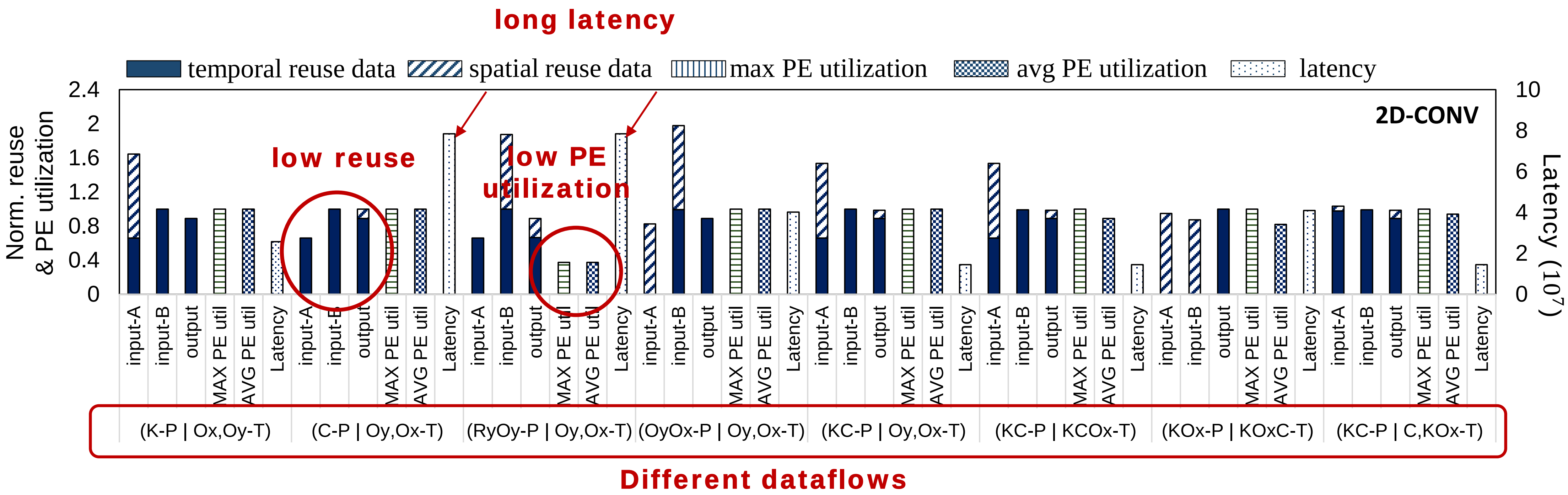
0.1s to evaluate a dataflow on a 2-core 2.50GHz Intel Core i5-7200U CPU

Complexity of interconnection topology

40min to finish the DSE of 2D-CONV

Number of loops

Metrics evaluation (2D-CONV)




```
{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
```

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

only keep inner-most two time dimensions

Notate dataflows

Tensor kernel	Dataflow	Relation-centric notation	Data-centric notation
	(KJ-P | K,IJK-T)	√	×
	(IK-P | K,IJK-T)	√	×

More expressive, more optimization opportunities

Tensor kernel	Dataflow	Relation-centric notation	Data-centric notation
2D-CONV	(KOX-P | OY,KOXC-T)	√	×
	(KC-P | OY,KCOX-T)	√	×
	(K-P | OX,OY-T)	√	√
	(RYOY-P | OY,OX-T)	√	√
	[illegible]	√	×

Support more tensor kernels

Tensor kernel	Dataflow	Relation-centric notation	Data-centric notation
Jacobi-2D	(I-P | I,J-T)	√	×
	(IJ-P | I,J-T)	√	×
MMc (Attention mechanism)	(IJ-P | J,IJL-T)	√	×
	(KJ-P | J,KJL-T)	√	×

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]}

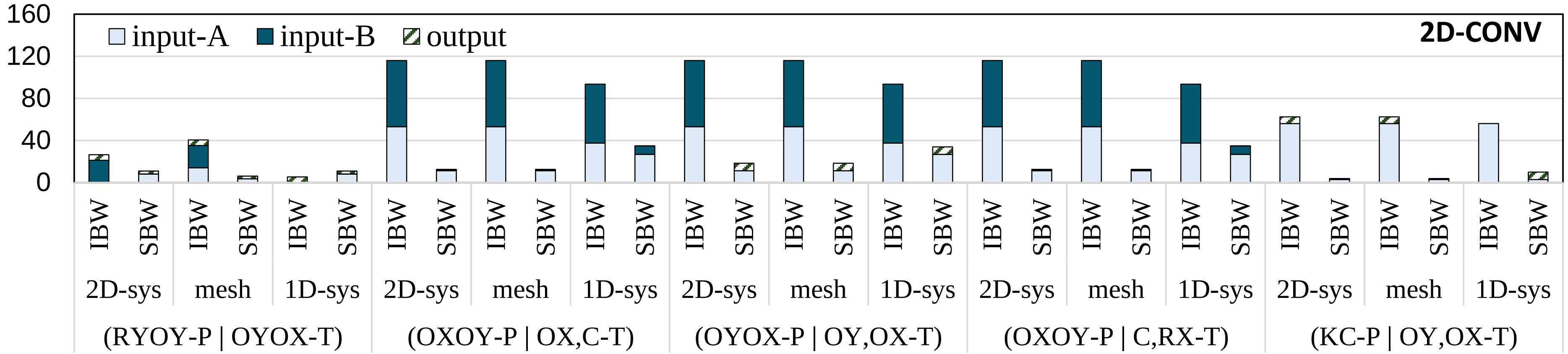
8x8 2D-systolic array:

{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]; 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]}

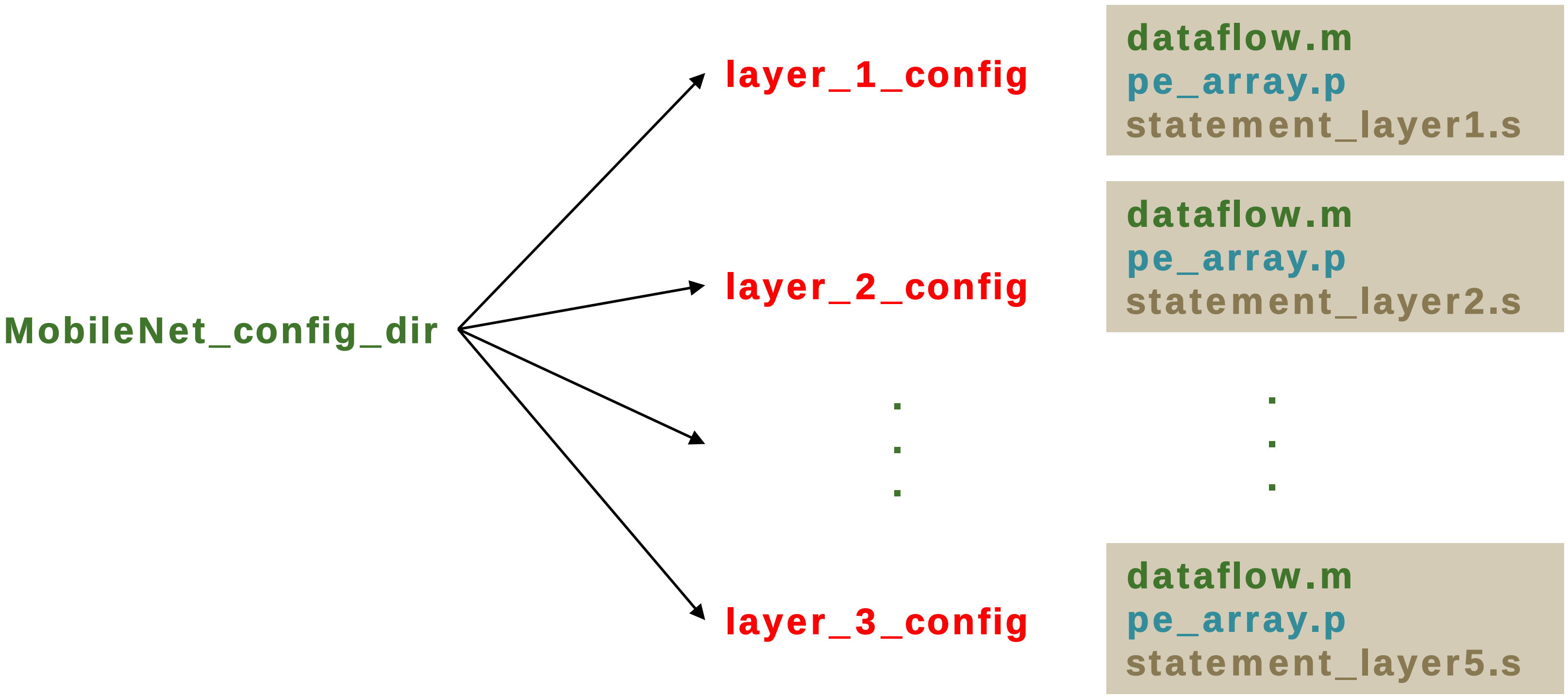
IBW: interconnection bandwidth. SBW: scratchpad bandwidth



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



Tutorial: model a dataflow (2/2)

STEP 2: run TENET

tenet –e ./network_example/MobileNet/config -d ./ network_example –o test.csv

TENET will analyze each layer in sequence

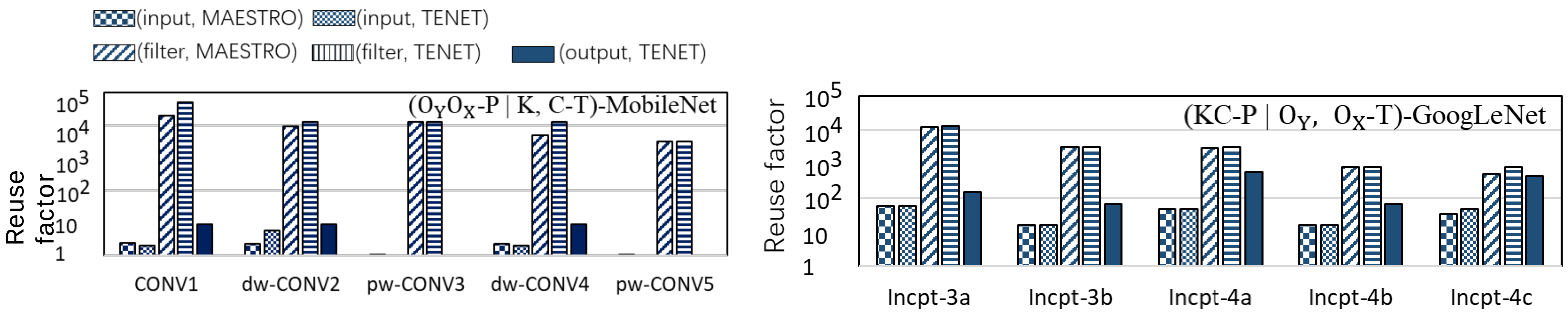
If no **--all, TENET only shows partial results**

```
Experiment experiment_5
Delay: In: 6424591; Out: 6422543; Com: 401408
Active PE Num: 64; Average: 64.00

Experiment experiment_1
Delay: In: 5438055; Out: 1204239; Com: 677376
Active PE Num: 64; Average: 64.00

Experiment experiment_4
Delay: In: 906463; Out: 200719; Com: 112896
Active PE Num: 64; Average: 64.00

Experiment experiment_3
Delay: In: 6423055; Out: 6422543; Com: 401408
Active PE Num: 64; Average: 64.00

Experiment experiment_2
Delay: In: 156887; Out: 100367; Com: 56448
Active PE Num: 64; Average: 64.00
```

test.csv

```
→  TENET git:(micro_tutorial) ✗ ./bin/tenet -h
Usage: tenet [options]
Options:
  -h, --help:                   Display this information.
  -e, --experiment <arg>:       Path of the expriment file or a directory.
                                If <arg> is a file, run the experiment file.
                                If <arg> is a directory, run all
                                experiment_? files inside of the directory.
                                When this argument is specified, -p, -m and
                                -s are ignored.
  -d, --experiment_dir <dir>:   Base directory for files in the experiment.
  -p, --pe_array <file>:        Path of the pe array file.
  -m, --mapping <file>:         Path of the mapping file.
  -s, --statement <file>:       Path of the statement file.
  -o, --output <file>:          Path of the output CSV file.
   --all:                       Print all infomation.
Contact semiwaker@pku.edu.cn for bugs
```

Network 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
- Document: https://tenet-docs.readthedocs.io/