AI Accelerator

영남대학교
차세대 컴퓨터 시스템 연구실
석사과정생 이원호

AI accelerator

인공지능 가속기에서의 논의사항

- FC(Fully Connected), CONV(Convolutional) 등과 같은 다양한 데이터 형태에 대한 최적화
- 다양한 데이터, 모델 형태에 따른 다양한 dataflow (data partitioning, scheduling 등)
- 메모리 용량
- PE간의 연결 방식

A Multi-Neural Network Acceleration Architecture

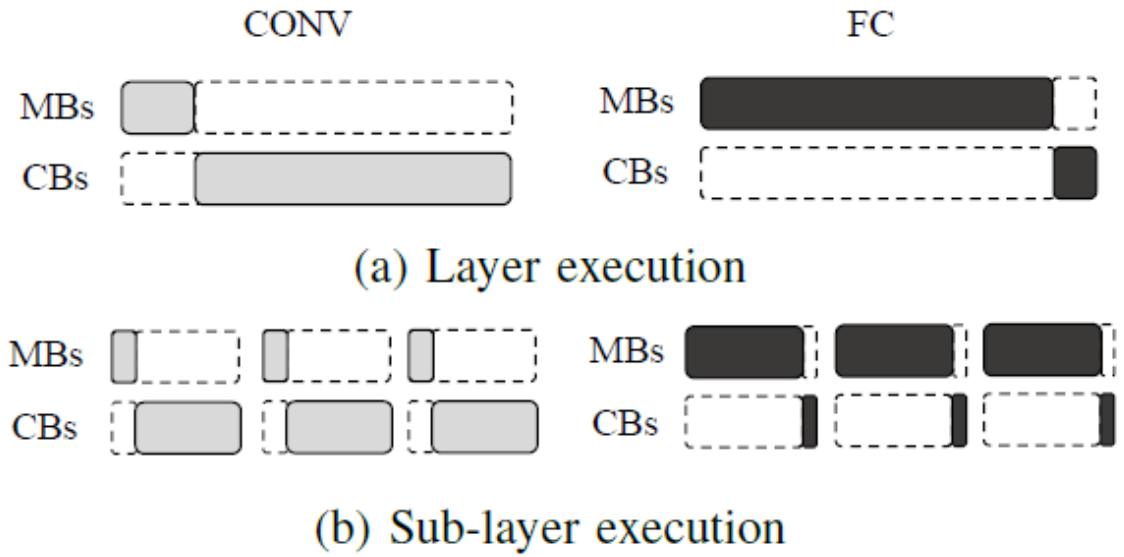
2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)

차세대 컴퓨터 시스템 연구실

이원호

기본 개념

- MB(Memory Block)
  - On-chip SRAM에 올라가는 단계
  - 메모리 영역
- CB(Compute Block)
  - 입력과 MB에 올라온 가중치를 함께 처리하는 단계
  - PE 영역
- 컴파일 단계에서 각 layer를 일정한 sub-layer로 나눔



문제점과 동기

- 클라우드 시스템에서 AI 가속기의 사용이 증가
- 여러 사용자가 AI 가속기를 이용하기 때문에 동시에 여러 신경망을 실행할 수 있도록 해야함

- 기존 AI 가속기들은 단일 신경망에 대해 최적화됨
  - 여러 신경망을 실행하려면, 순차적으로 실행하기 때문에, throughput성능이 떨어짐
- 다양한 형태의 신경망
  - Fully-Connected와 같이 memory-intensive한 신경망, CONV와 같은 compute-intensive한 신경망 등 신경망 별로 다양한 특성을 가짐
- 메모리 접근과 계산의 불균형
  - 다양한 형태의 신경망으로 인해, 자원의 사용량이 달라, 특정 자원의 고갈이 발생할 수 있음

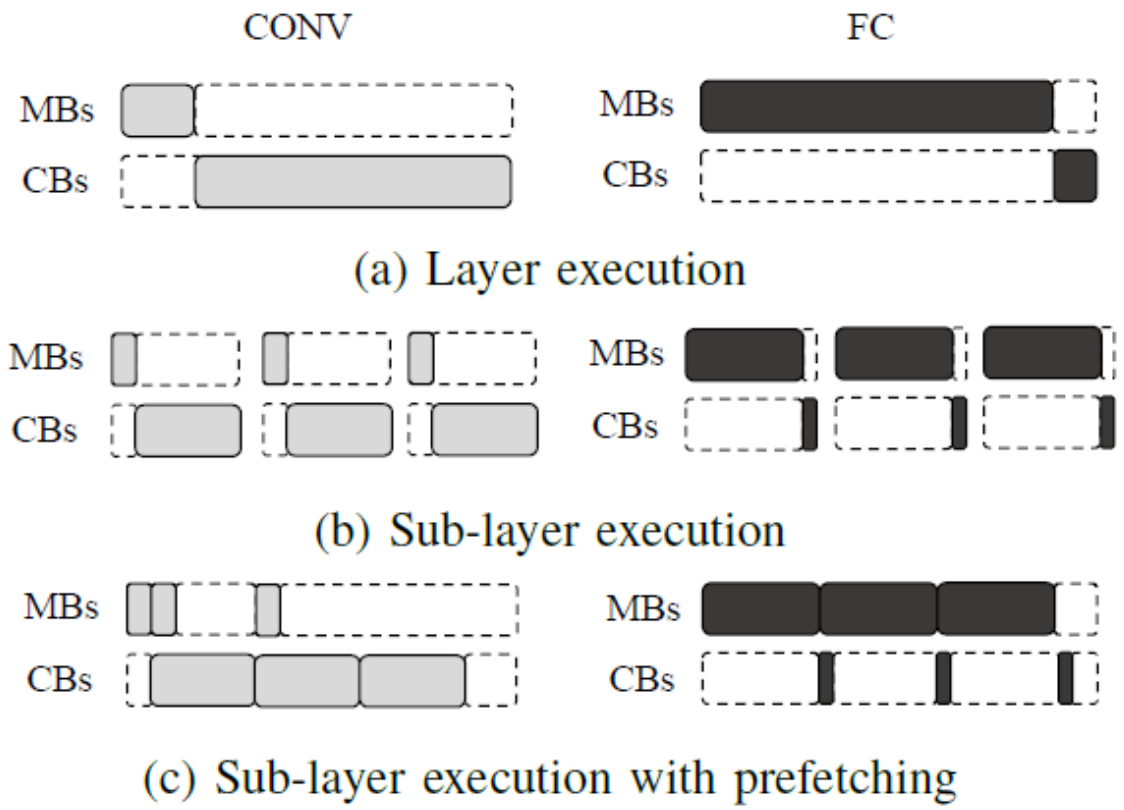
제안 사항

- Memory Block Prefetching
  - 부하 균형을 맞추기 위함 (Memory Idleness)
- Compute Block Merging
  - 부하 균형을 맞추기 위함 (PE Idleness)
- Memory Block Eviction
  - On-chip 메모리 사용을 최소화함 (SRAM capacity)

Memory Block Prefetching

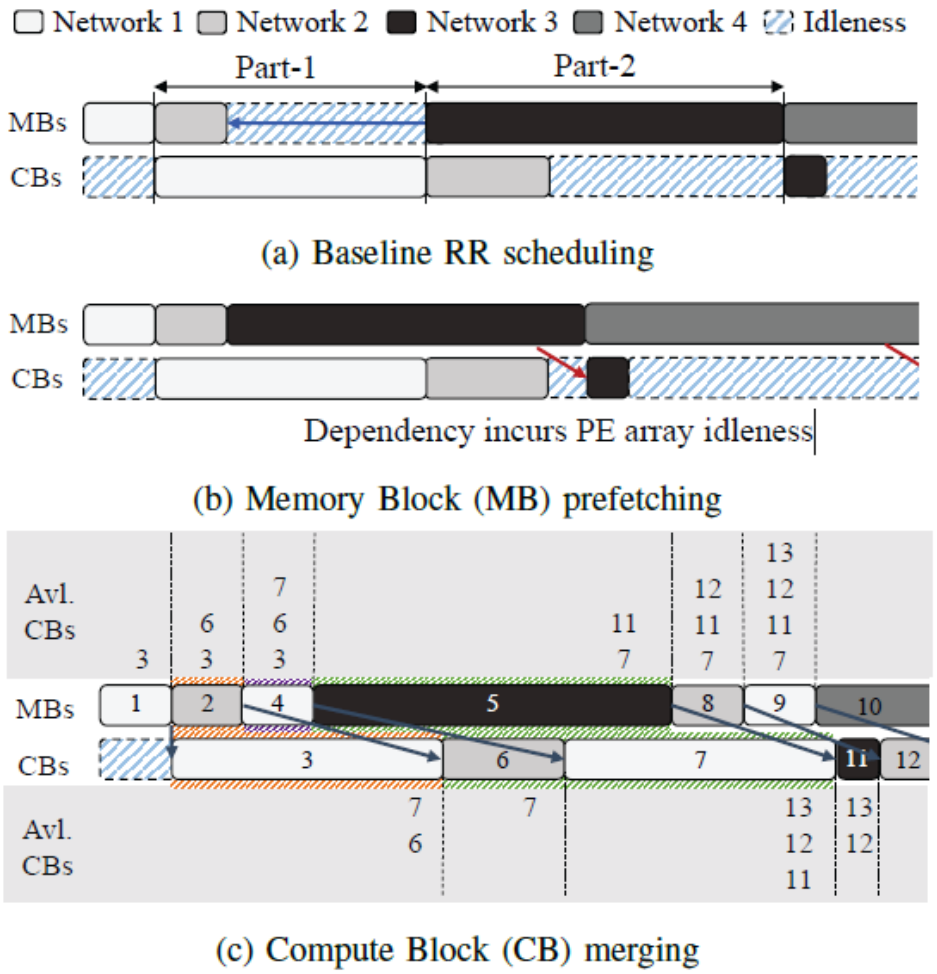
- 기존 가속기에서는 CB가 진행되는 동안 MB 과정이 진행되지 않음

- CB 과정 중에 다음 MB를 진행해 MB Idleness를 줄임



Compute Block Merging

- MB가 CB보다 큰 경우에 sub-layer간의 의존성으로 인해 PE의 성능이 떨어지게 됨
- 다음 MB를 올리기에 충분한 CB 실행 시간을 갖는 CB 후보들을 추적하고 있음
- PE의 Idle time을 줄일 수 있음

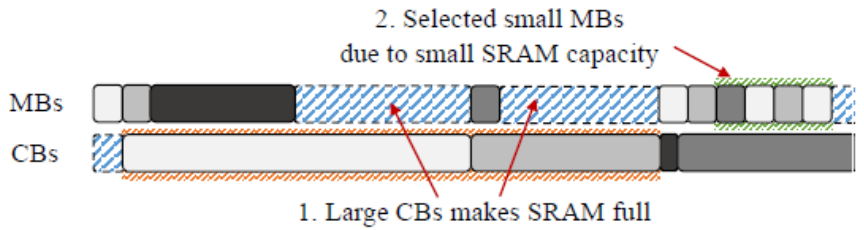


Memory Block Eviction

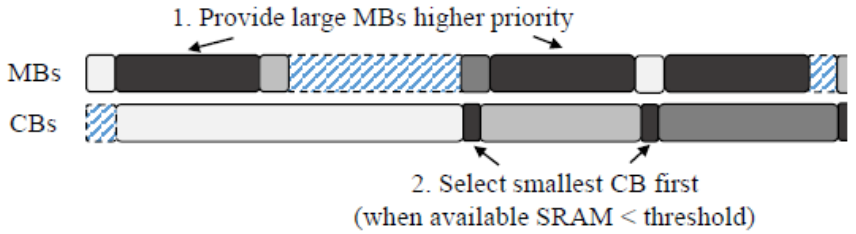
- CB의 실행시간이 긴 경우 SRAM에 MB가 지속적으로 prefetching이 진행되면, SRAM 부족으로 MB의 idle time이 증가함

1. 작은 CB를 먼저 처리해 SRAM의 확보

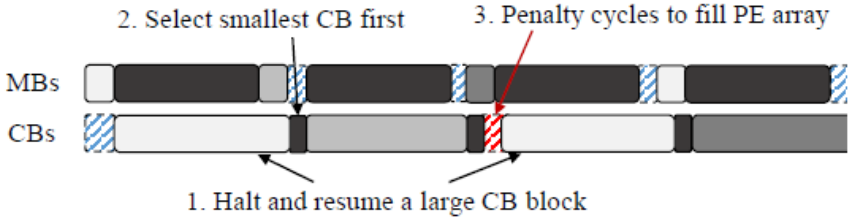
2. CB block을 작은 단위로 나눠서 처리해 SRAM을 확보



(a) Resource idleness under SRAM capacity shortage



(b) Priority mechanism of the early MB eviction



(c) Split mechanism of the early MB Eviction

Fig. 13: SRAM capacity aware scheduling

요약

- 하드웨어의 idle time을 줄이기 위한 시도

Adapt-Flow: A Flexible DNN Accelerator Architecture for Heterogeneous Dataflow Implementation

GLSVLSI '22: Proceedings of the Great Lakes Symposium on VLSI 2022

차세대 컴퓨터 시스템 연구실

이원호

문제점과 동기

- Layer별로 그에 적합한 dataflow 전략이 존재하지만 기존 시스템은 하드웨어에 따라 고정된 dataflow 전략

- Dataflow 전략에 따라 최대 51%의 DRAM 접근량의 차이가 있음

제안 사항

- A flexible interconnect
  - 다양한 dataflow에서 요구하는 traffic 패턴을 동적으로 지원 가능
  - Clos network의 특성을 활용(unicast, multicast, broadcast 지원)

- A dataflow selection algorithm
  - 성능 향상을 목적으로 DNN layer에 대해 최적의 dataflow 전략을 취함

- A dataflow mapping technique
  - Flex interconnect에 맞는 dataflow를 효과적으로 연결

Flexible interconnect

- Weight stationary
  - Weight에 대한 메모리 접근을 줄임
- Output stationary
  - Partial sum에 대한 메모리 접근을 줄임
- Input stationary
  - Input fmap(activation)에 대한 메모리 접근을 줄임

Buffer에서 해당 값을 지니고 있음

Table 1: Traffic Pattern of Different Dataflows

Dataflow	Weight	Input Activation	Psum
WS	Unicast	Multicast	Unicast
OS	Broadcast	Unicast	Unicast
IS	Multicast	Unicast	Unicast

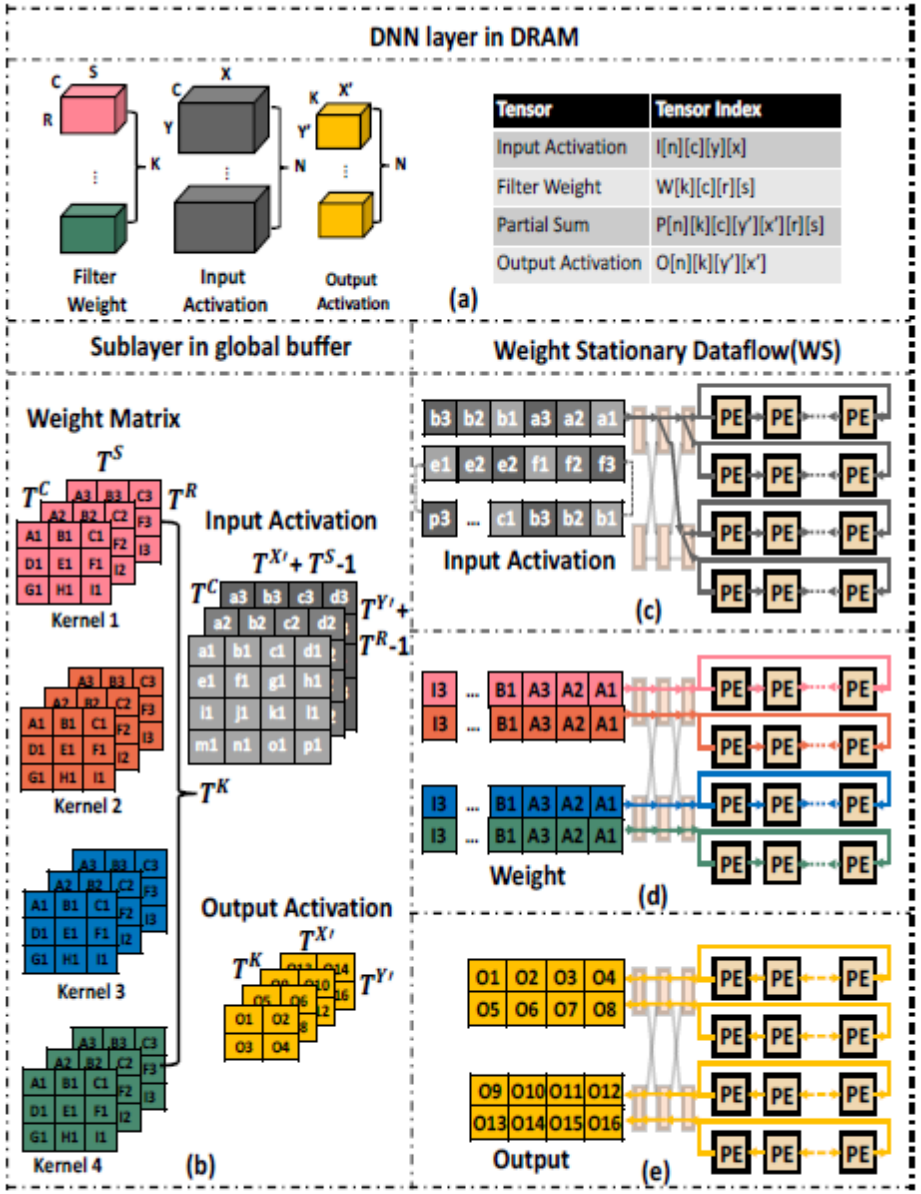


Figure 2: An example of our proposed accelerator processing a simple tiled convolutional layer using WS dataflow.

Dataflow selection algorithm

- 입력 특징값(input feature map)
- 출력 특징값(partial sum)
- 가중치(weight)

위의 세 가지 모두에 관해 DRAM 접근이 가장 적은 dataflow를 선택

Algorithm : Dataflow Selection

Input :Features of n dataflows: $DF_i, i \in \{1, \cdots, n\}$
Input :Capacity of global buffer: GLB
Input :Data dimension of DNN layer
Output :Optimal dataflow

```
1  begin
2      DA ← {} ; // The DRAM access volume for all
         dataflow.
3      /* Iterating all dataflows                    */
4      for i = 1 to n do
5          Calculate V_i^d, d ∈ {wt, ifmap, psum} ; // Data
             volume involved in each invocation
6          Calculate R_i^d, d ∈ {wt, ifmap, psum} ; // Number
             of invocations
7          Calculate G_i ; // Global buffer capacity
             requirement
8          if G_i ≤ GLB then
9              /* Calculate the DRAM access volume if
                 satisfying global buffer
                 requirement.                         */
10             DA_i ← Σ_d V_i^d × R_i^d, d ∈ {wt, ifmap, psum}
11         else
12             /* Otherwise don't consider by
                 assigning as infinity.               */
13             DA_i ← ∞
14         end
15     end
16     /* Return the dataflow with minimum DRAM
            access volume.                            */
17     return {DF_i | DA_i = min{DA}}
18 end
```

Figure 3: Dataflow Selection Algorithm.

Dataflow mapping technique

- WS 최적화 가속기를 예시로 모든 dataflow에 대해 적용할 수 있도록 mapping 기술을 제안
- 기존에는 unicast, broadcast 모두 row-wise bus를 활용
- 각 row에 대해 broadcast를 활용하기에는 ring topology가 비효율적

- Broadcast 통신을 쓰는 연산 (spatial optimization)을 요구하는 경우 같은 행에 매핑
- Unicast 통신을 쓰는 연산 (temporal optimization)을 요구하는 경우 같은 열에 매핑

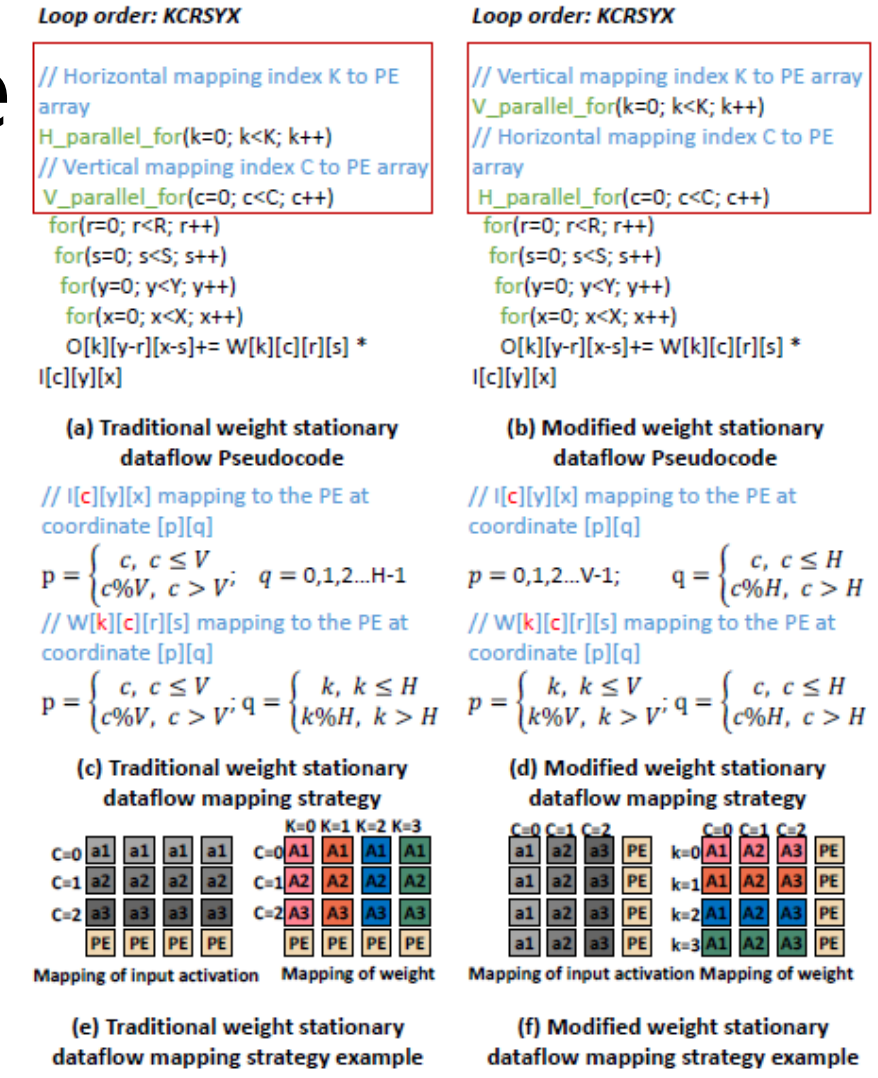


Figure 4: The differences between proposed mapping strategy and traditional mapping strategy for weight stationary dataflow.

요약

- 신경망의 형태에 따라 적합한 dataflow를 사용해 DRAM의 접근 횟수를 줄이는 시도

- 또한 그에 적합한 dataflow 매핑 방식 고안

Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach

MICRO '52: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019

차세대 컴퓨터 시스템 연구실

이원호

문제점과 동기

- Dataflow의 결과와 선택에 대한 직관적 이해와 HW/SW 동시 최적화 설계 영역을 탐색하기 위한 도구와 방법론의 부족

- 이에 대한 탐색 방법론으로 기존의 compute-centric notation이 아닌 data-centric notation을 제안함

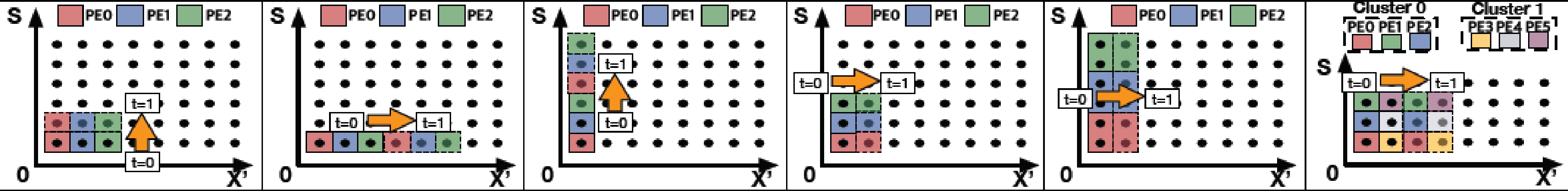
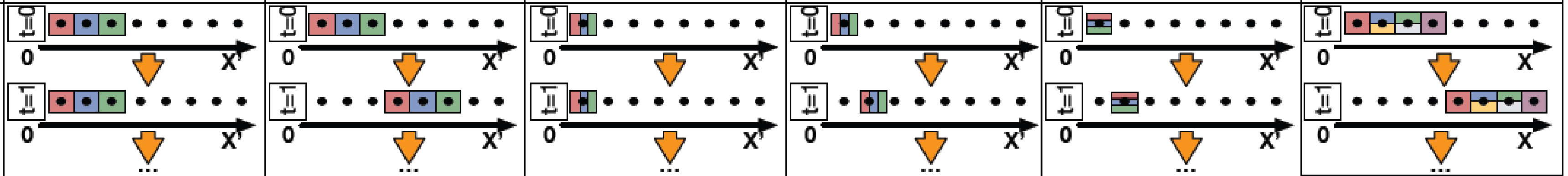
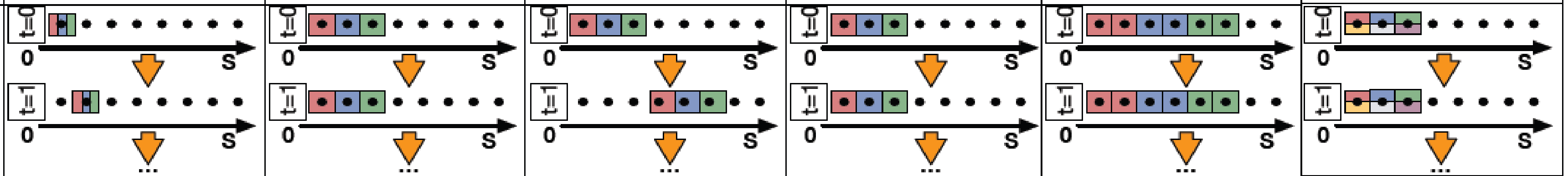
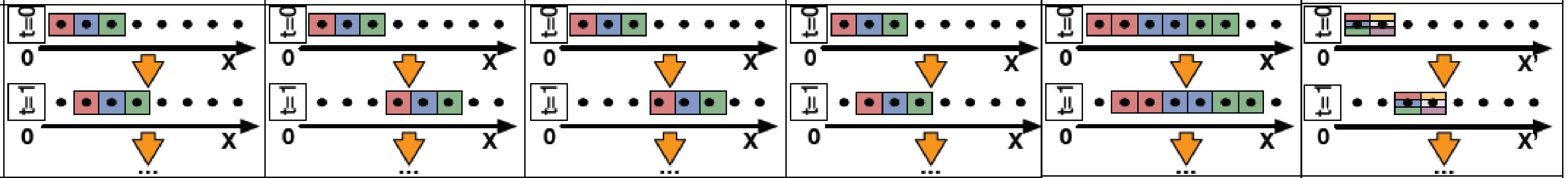
- 탐색 도구로 MAESTRO 분석 도구를 제안함

Data-centric notation

- 기존의 compute-centric notation은 loop-nest 표현을 통해 data reuse를 추론함 (직관성이 부족함)
- Data-centric notation을 통해 넓은 범위의 data reuse를 표현할 수 있음 (spatial, temporal, spatial-temporal)

Data-centric notation

1. Spatial map(size, offset) α
   PE에 대해 α 의 데이터 차원 분할을 구체화함
   Size = 각 PE에 대해 α차원에서 매핑된 인덱스의 수
   Offset = 연속적인 PE에 대해 α의 시작 지표의 변화
2. Temporal map(size, offset) α
   한 PE에서 각 타임 스탭마다 α 차원 분할을 구체화함
   각 타임 스탭에서 PE에 대해 매핑된 차원 지표의 조각은 같다.
3. Data movement order
   Dataflow 구체화에서 공간적, 시간적 매핑 순서가 가리키는 데이터 이동 순서
4. Clusters
   다차원 데이터를 활용하기 위해, PE를 그룹화하는 것

Dataflow ID	A	B	C	D	E	F
Mapping	SpatialMap (1, 1) X' TemporalMap (1, 1) S	TemporalMap (1, 1) S SpatialMap (1, 1) X'	TemporalMap (1, 1) X' SpatialMap (1, 1) S	SpatialMap (1, 1) S TemporalMap (1, 1) X'	SpatialMap (2, 2) S TemporalMap (1, 1) X'	TemporalMap (3, 3) S SpatialMap (1, 1) X' Cluster (3) SpatialMap (1, 1) S TemporalMap (1, 1) X'
Iteration Space (Partial sums)						
Output Featuremap Data Space						
Filter Weight Data Space						
Input Featuremap Data Space						
Temporal Reuse	- Temporal reduction of outputs (Output stationary)	- Temporal multicast of weights (Weight stationary)	- Temporal reduction of outputs (Output stationary)	- Temporal multicast of weights (Weight stationary)	- Temporal multicast of weights (Weight stationary) - Partial temporal multicast of inputs (e.g., X =3 is used by PE1 over t=0 and 1)	- Temporal multicast of weights (Weight stationary)
Spatial Reuse	- Spatial multicast of filter weights	- Spatial multicast of filter weights	- Spatial reduction of outputs	- Spatial reduction of outputs	- Spatial reduction of outputs	- Spatial reduction of outputs
Informal Dataflow Name	Output-Stationary	Weight Stationary	Collaborative Output-Stationary	Collaborative Weight-Stationary	Tiled Collaborative Weight-Stationary	Clustered Tiled Collaborative Weight-Stationary

요약

- 구조화되지 않은 dataflow에 대해 구조적인 data-centric notation이라는 방법론을 제시함

- Data reuse에 대한 이해와 DNN dataflow를 정형화된 형태로 구체화함

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

2021 ACM/IEEE 87th Annual International Symposium on Computer Architecture (ISCA)

차세대 컴퓨터 시스템 연구실

이원호

문제점과 동기

- 기존의 compute-centric, data-centric이 직관적이지 않다는 사실
- Compute-centric은 loop instance에 대해서만 표현하기 때문에 추가적인 loop 변환을 통해 data transfer, data reuse를 표현해야 함
- Data-centric notation은 이를 해결했지만, tensor의 차원을 표현하지 못한다.

Limitation of Compute-centric and Data-centric notation

- Compute-centric notation의 경우 data reuse, locality를 위해 좌측 그림처럼 loop instance에 대해 추가적인 변환이 필요

- Data-centric notation의 경우 실제 data reuse와 예측한 data reuse가 다름

- 두 notation 모두 Rectangle data access만 가능함

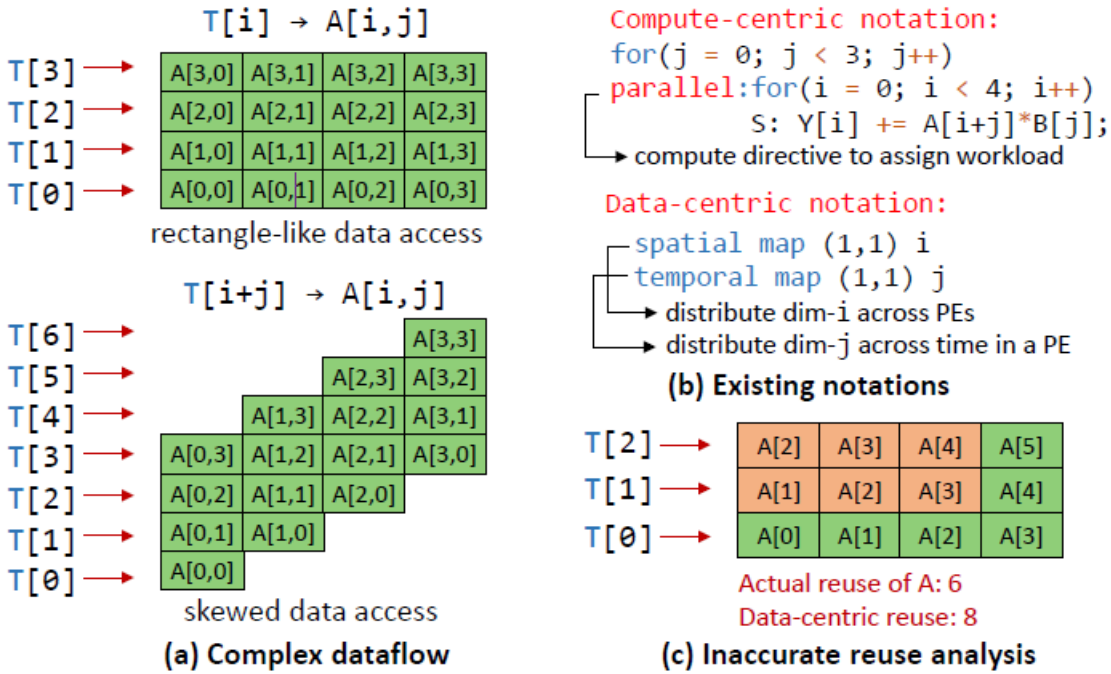


Fig. 1: Limitation of compute-centric and data-centric notation.

Relation-centric notation

- 관계를 통해 tensor 연산을 위한 dataflow를 설명함

1. Loop instance와 해당 연산을 수행하는 PE와의 관계
2. Loop instance와 PE에서의 계산 순서의 관계
3. PE와 그와 관련해 할당된 tensor element 사이의 관계
4. 상호 연결된 PE들 간의 관계

관계1,2는 언제, 어디서 loop instance가 실행되는가를 결정

관계 3은 언제, 어디서 tensor element가 접근되는가를 결정

관계4는 tensor elements들이 어떻게 PE를 순회하는가를 설명 (systolic array, reduction tree)

Relation-centric notation

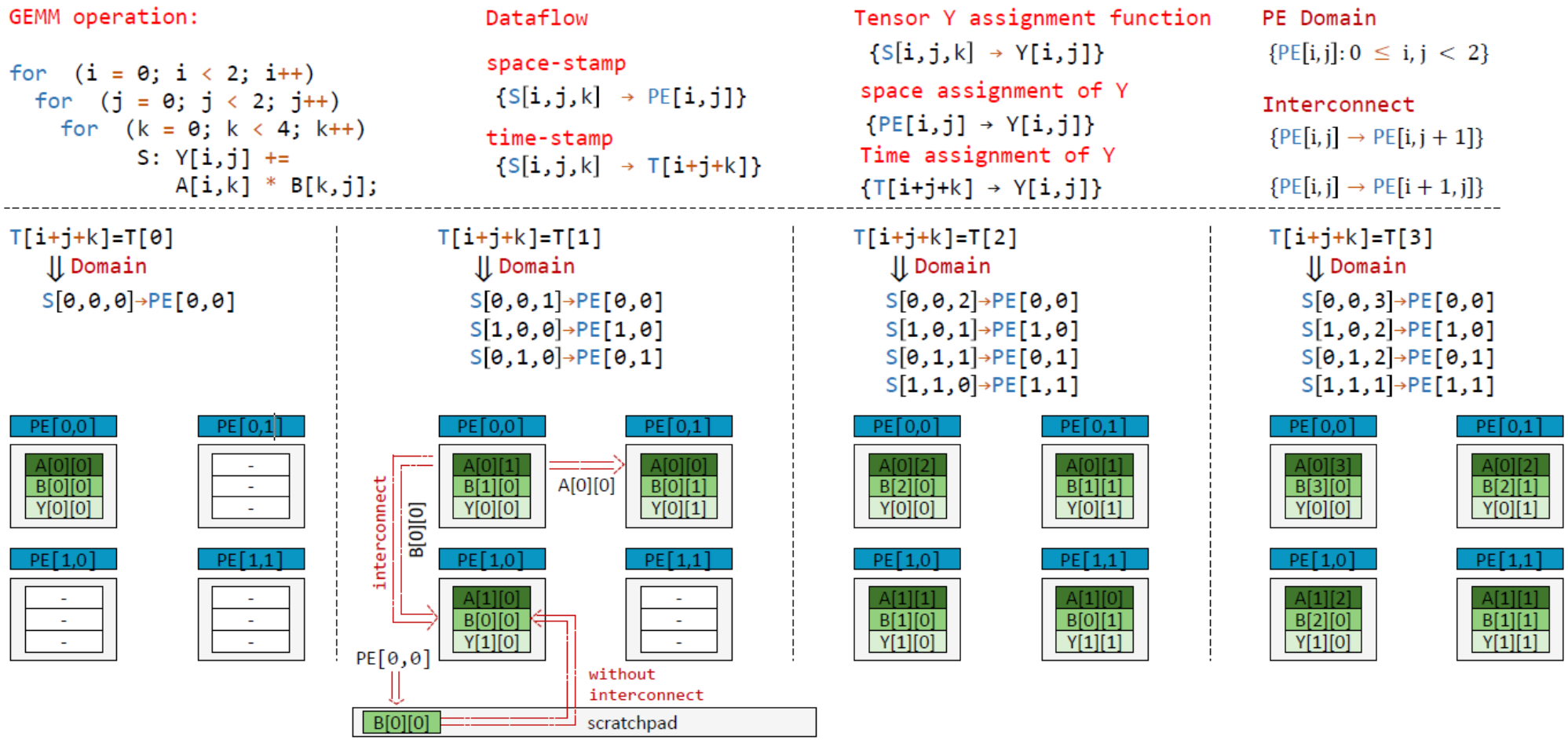


Fig. 3: Analyzing the dataflow of matrix multiplication on a 2×2 PE array using the relation-centric notation.

요약

- 기존 dataflow notation의 문제점을 기술
- 관계라는 일관적인 방법을 통해 dataflow에 대해 구조적인 접근이 가능해짐

Memory-Computing Decoupling: A DNN Multitasking Accelerator With Adaptive Data Arrangement

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2022

차세대 컴퓨터 시스템 연구실

이원호

문제점과 동기

- 각 layer의 subtask들은 신경망의 크기와 모양에 따라 이종성을 가지기 때문에 각자 특별한 dataflow를 요구하기 때문에, DNN 가속기들은 다양한 dataflow를 요구함

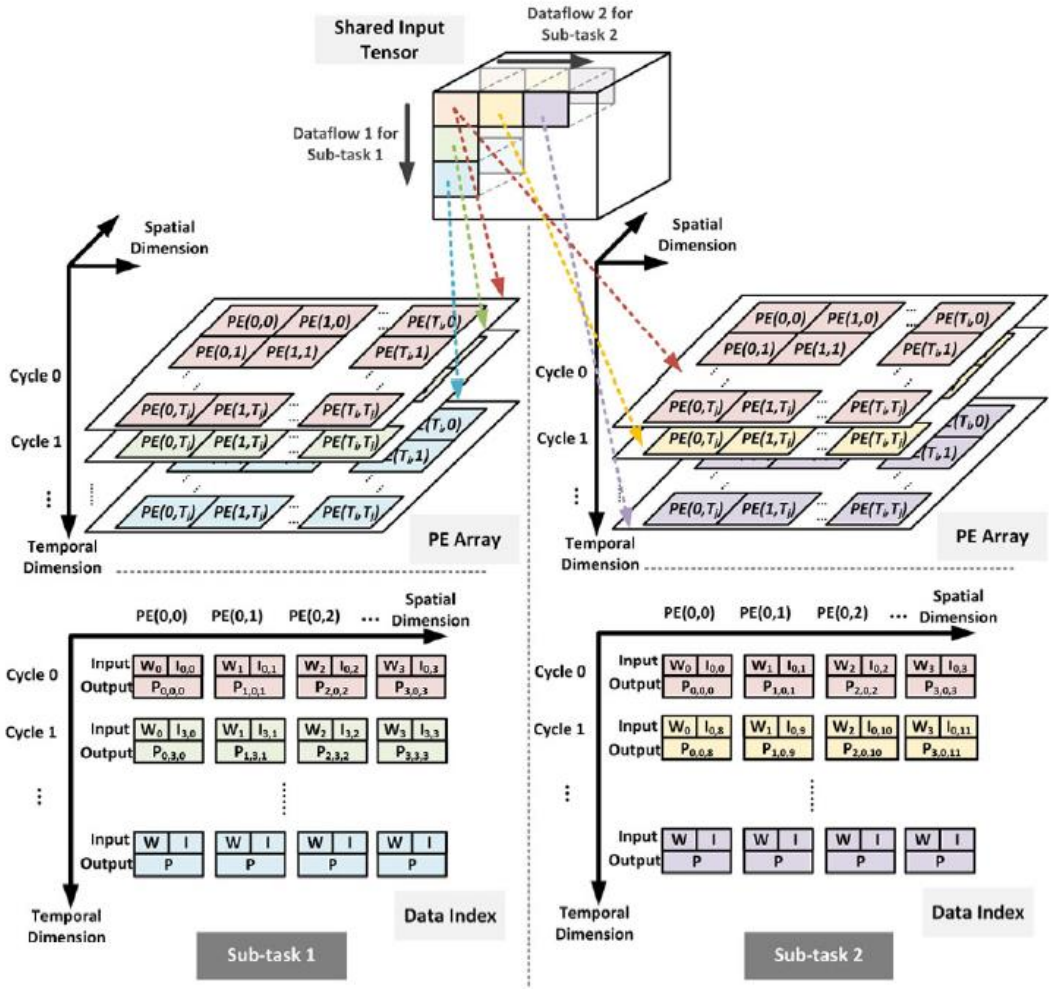
- Dataflow 접근 방식의 병렬 처리(다차원 구조)와 선형적인 메모리 구조(1차원 구조) 간의 차원 불균형으로 인해 성능이 낮아짐

- 여러 DNN 작업이 부분적인 특징이나 가중치를 공유할 때, 이런 불균형이 악화됨

- 불균형 -> 메모리에서 데이터 공급을 느리게 만듦

Dimension mismatch

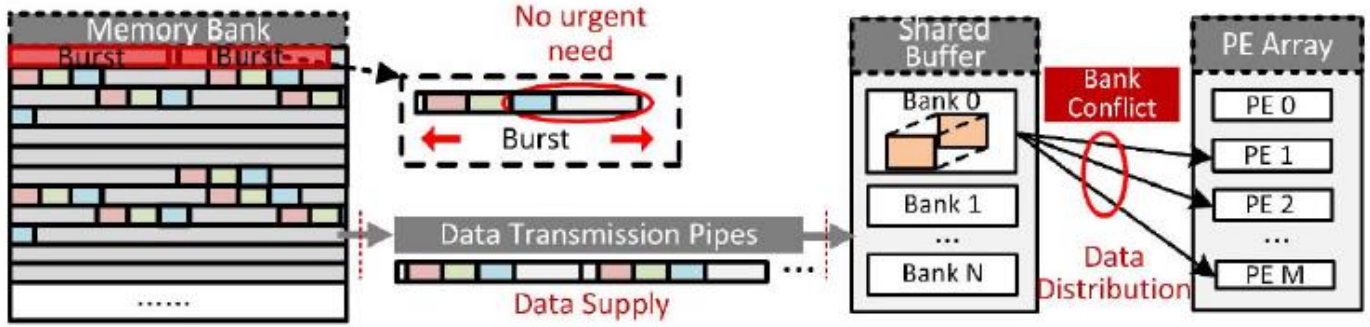
- 메모리와 계산 유닛들이 요구하는 데이터 차원이 다르기 때문에, 둘 중하나가 데이터 관리에 대해 지배적인 영향을 가지게 됨
- PE array가 이를 관리하는 경우가 많음
- 두 DNN이 공유를 하면 목적에 따라 서로 다른 dataflow를 가짐
- 다른 dataflow로 인해 데이터의 색인과 할당이 매우 달라짐



Dimension mismatch

- Memory 주소 공간이 연속적이지 않을 수 있기 때문에, data space 와 time locality에 대해 data access conflicts가 발생할 수 있다.

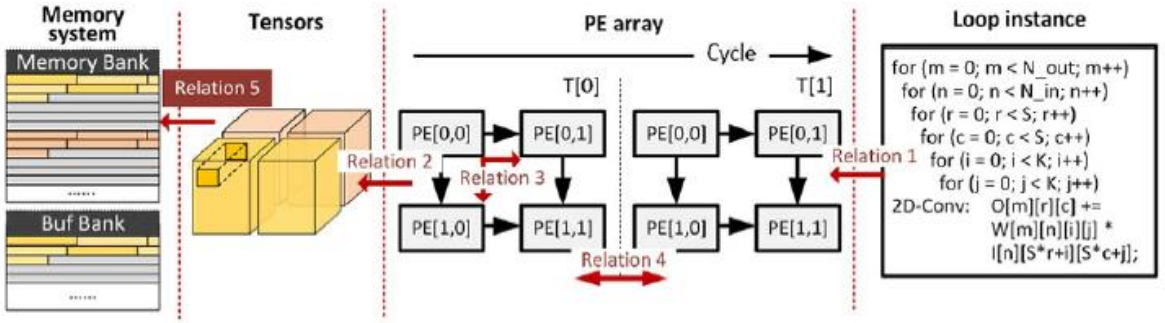
- 메모리와 연산 처리 간 차원 불균형을 맞추기 위해, 이를 분할하고 그 사이에 tensor dimension conversion unit을 추가한다.



Dataflow notation

- 기존 relation-centric notation에서는 언제, 어떤 관계에 대해서만 설명하고, 메모리 접근에 대한 설명이 없으며 이에 대한 정량적인 해석이 부족함.

- 그런 이유로 메모리 접근에 대한 5번째 관계를 설정해 이를 해결



요약

- 기존 시스템의 문제점에 대해 차원 불일치의 문제점을 기술하고, 이가 시스템에 미치는 영향을 서술함

- 기존 dataflow notation에서 빈약한 메모리 접근에 대한 부분을 추가해 좀 더 직관적인 방법론을 제시함

최종 요약

- 클라우드 시스템에서의 사용과 멀티 모달 모델의 수요로 인해 Parallelism 에 대한 논의가 많이 발의되고 있음
- HW적인 측면에서 mapping 방식이나, 연결 방식에 대한 논의
- SW적인 측면에서 data partitioning, scheduling에 대한 논의
- 전체적인 dataflow를 설명하는 방법론에 대한 대한 논의
- 다양한 dataflow와 모델에 대해 하나의 방식에 대한 최적화가 아닌 여러 dataflow를 수용할 수 있는 동적인 방식에 대한 논의

Reference

- A Multi-Neural Network Acceleration Architecture,2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)
- Adapt-Flow: A Flexible DNN Accelerator Architecture for Heterogeneous Dataflow Implementation, GLSVLSI '22: Proceedings of the Great Lakes Symposium on VLSI 2022
- Understanding Reuse, Performance, and Hardware Cost of DNN Dataflows: A Data-Centric Approach, MICRO '52: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019
- TENET: A Framework for Modeling Tensor Dataflow Based on Relation-centric Notation, 2021 ACM/IEEE 87th Annual International Symposium on Computer Architecture (ISCA)
- Memory-Computing Decoupling: A DNN Multitasking Accelerator With Adaptive Data Arrangement, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2022