6.5930

Hardware Architecture for Deep Learning

Mapping - Partitioning

February 18, 2026

Joel Emer and Vivienne Sze

Massachusetts Institute of Technology

Electrical Engineering and Computer Science

Partitioning

Objectives of Partitioning

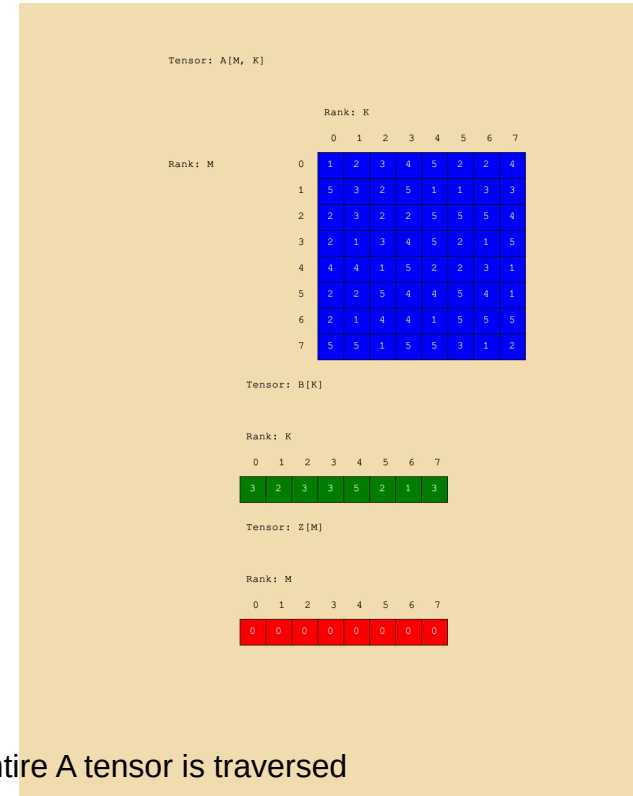
- Reduce the reuse distance between accesses to a tensor value so that the value can be retained in a buffer
- Indentifying separate sets of data for computing in parallel

Unpartitioned Matrix-Vector Computation

$$Z_m = A_{k,m} \times B_k$$

```
# Matrix-vector multiply

for m in range(M1):
    for k in range(K):
        Z[m] += A[k,m] * B[k]
```



Note how for each output value the entire A tensor is traversed

Partitioning a Vector

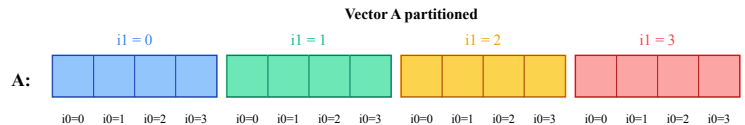
Unpartitioned

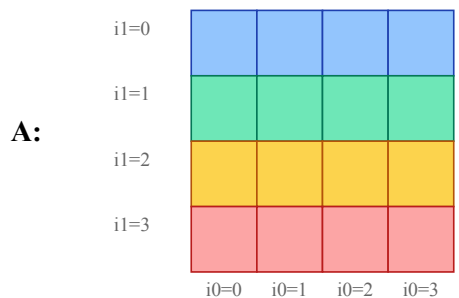
$$A_i$$

Partitioned

$$A_{i1,i0}$$

Where: $i \equiv i1 \times I0 + i0$





Partitioning always adds a rank

Note: $I = I1 \times I0$.

Partitioning a Matrix

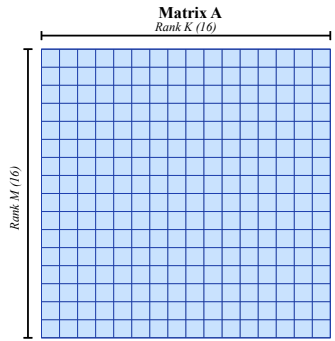
Unpartitioned

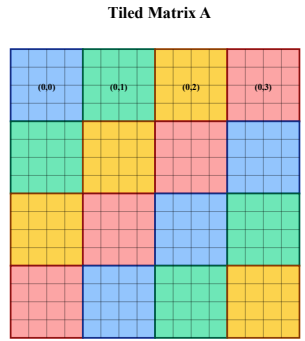
$$A_{k,m}$$

Partitioned

$$A_{k1,k0,m1,m0}$$

Where: $k \equiv k1 \times K0 + k0$
and $m \equiv m1 \times M0 + m0$





Partitioning always adds a rank???

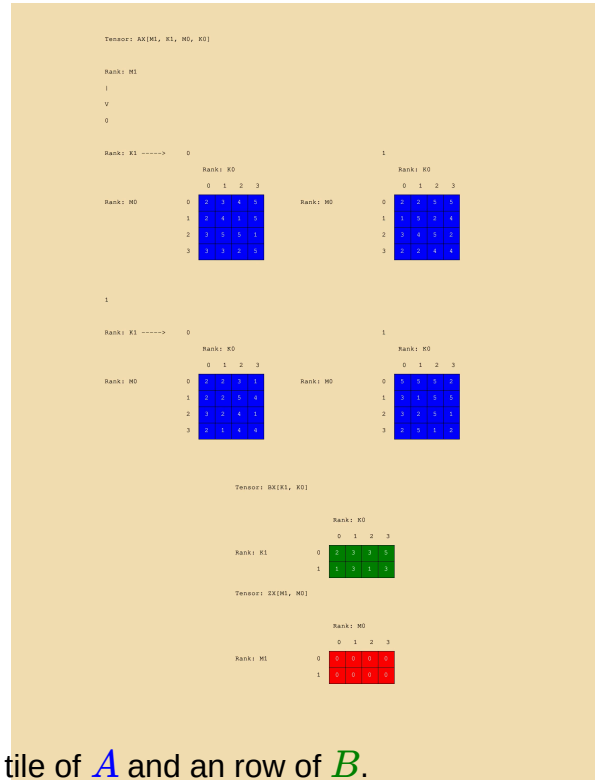
Note: $K = K1 \times K0$ and $M = M1 \times M0$

Partitioned Matrix-Vector Computation

$$A_{k1,k0,m1,m0}$$
$$B_{k1,k0}$$
$$Z_{m1,m0}$$

$$Z_{m1,m0} = A_{k1,k0,m1,m0} \times B_{k1,k0}$$

```
# Matrix-vector multiply
for m1 in range(M1):
    for k1 in range(K1):
        for m0 in range(M0):
            for k0 in range(K0):
                Z[m1,m0] += A[k1,k0,m1,m0] * B[k1,k
```



Note how computation stays in an M0 x K0 tile of A and an row of B.

Partitioning for Parallelism

This example shows partitioning of element-wise multiplication to achieve parallelism

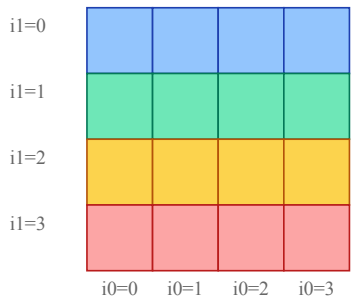
Unpartitioned

$$Z_i = A_i \times B_i$$

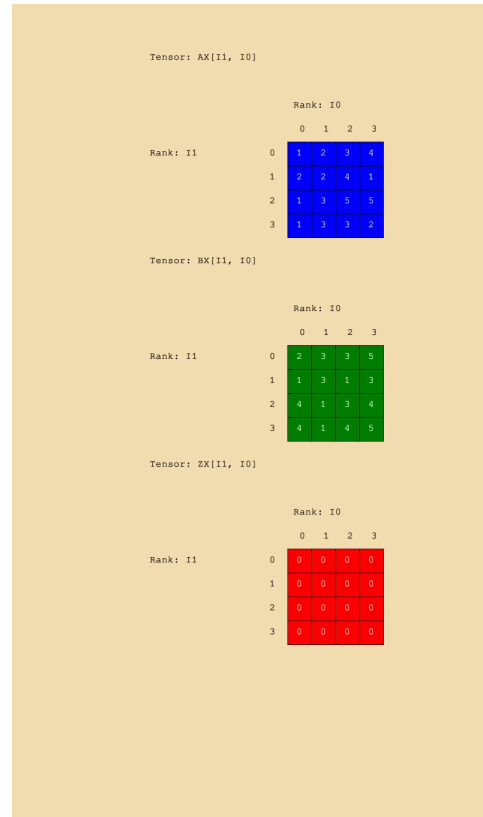
Partitioned

$$Z_{i1,i0} = A_{i1,i0} \times B_{i1,i0}$$

A:




```
spatial-for i1 in [0,I1):
    for i0 in [0,I0):
       z[i1,i0] = a[i1,i0] * b[i1,i0]
```



Distributed Matrix Multiply

Objective

The obective of the this mapping is to perform a distributed matrix multiply where only portions of the operands fit in the storage of each processing element. The basic computation is:

$$Z_{m,n} = A_{k,m} \times B_{k,n}$$

This implemenation is based on the algorithm used in [Spector, ThunderKittens, ICLR 2025]

Overview

The basic idea of this dense matrix-matrix multiply is to facilitate the implementation of a parallel dataflow using the following Einsum on partitioned A, B and Z tensors:

$$Z_{m1,m0,n} = A_{k1,k0,m1,m0} \times B_{k1,k0,n}$$

Which is equivalent to this computation with flattened ranks:

$$Z_{(m1,m0),n} = A_{(k1,k0),(m1,m0)} \times B_{(k1,k0),n}$$

Then linearizing the tuples results in:

$$Z_{(m1 \times M0 + m0),n} = A_{(k1 \times K0 + k0),(m1 \times M0 + m0)} \times B_{(k1 \times K0 + k0),n}$$

Substitution results in the original matrix multiply

$$Z_{m,n} = A_{k,m} \times B_{k,n}$$

Delayed Reduction

Original computation

$$Z_{m1,m0,n} = A_{k1,k0,m1,m0} \times B_{k1,k0,n}$$

Delayed reduction variant

1. Move $k1$ to the left hand side to delay the reduction

$$ZT_{k1,m1,m0,n} = A_{k1,k0,m1,m0} \times B_{k1,k0,n}$$

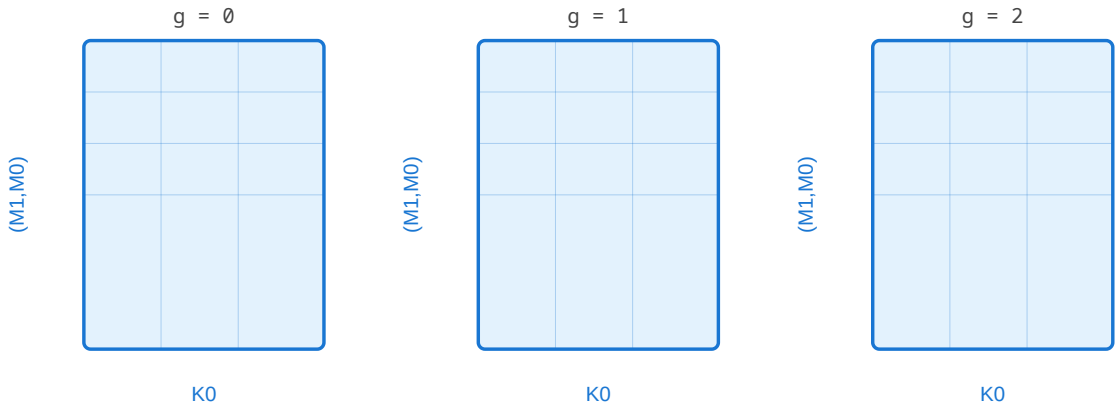
2. Do the reduction

$$Z_{m1,m0,n} = ZT_{k1,m1,m0,n}$$

Distributed partitions

The partitions of the operands can be distributed to a set of PEs. So that we will ultimately use G PEs (GPUs in the original work) to perform the matrix multiplication

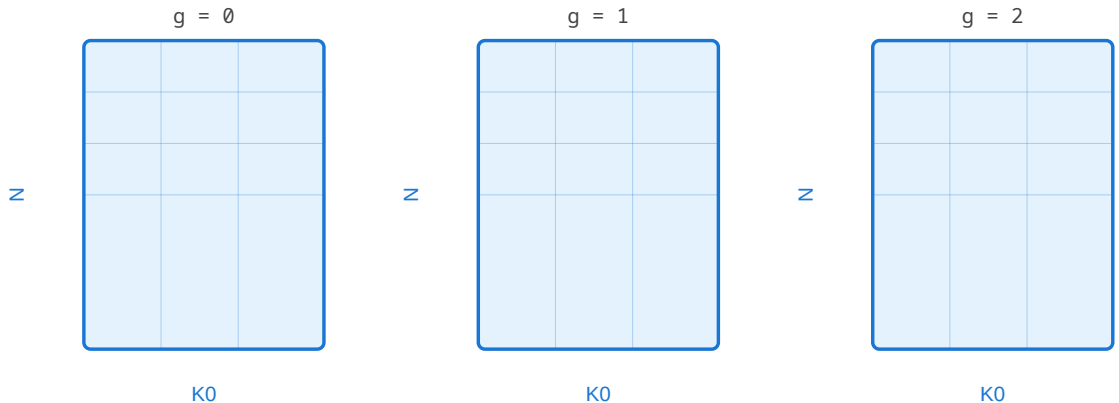
Distributed A - $AD^{G,K0,M1,M0}$



Structure:

- G matrices (one per K1 value)
- Each local matrix has shape: K0 × (M1 × M0)
- Rows indexed by flattened (m1, m0) pairs
- Columns indexed by k0

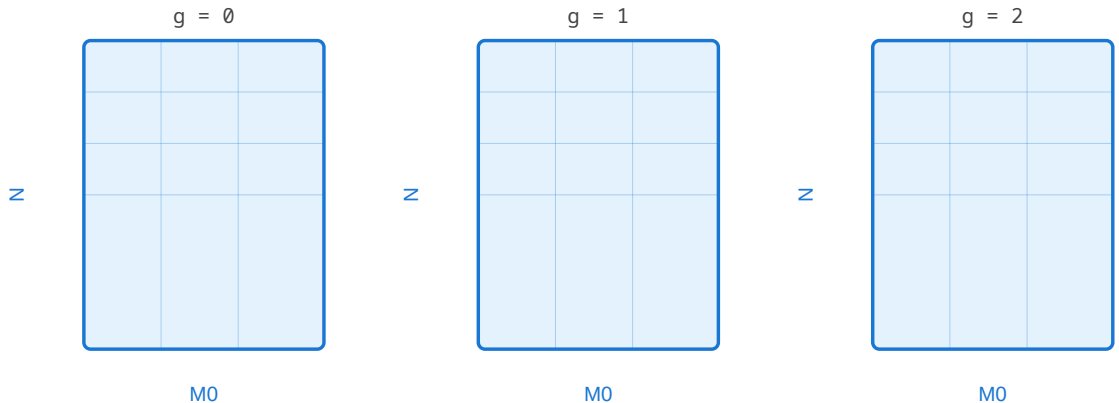
Distributed B - $BD^{G,K0,N}$



Structure:

- G matrices (one per K1 value)
- Each local matrix has shape: K0 × N
- Rows indexed by n
- Columns indexed by k0

Distributed Z - $ZD^{G,M0,N}$



Structure:

- G matrices (one per M1 value)
- Each local matrix has shape: M0 × N
- Rows indexed by n
- Columns indexed by m0

Einsums!

Distribution

$$AD^{G,K0,M1,M0}_{g,k0,m1,m0} = A^{K1,M1,K0,M0}_{g,m1,k0,m0}$$

$$BD^{G,K0,N}_{g,k0,n} = B^{K1,K0,N}_{g,k0,n}$$

Main Computation

$$ZL^{G,M1,M0,N}_{g,m1,m0,n} = AD_{g,k0,m1,m0} \times BD_{g,k0,n}$$

$$ZD^{G,M0,N}_{g,m0,n} = ZL_{h,g,m0,n}$$

Finalization

$$Z^{M1,M0,N}_{g,m0,n} = ZD_{g,m0,n}$$

Approach Summary

The approach is to partition the input tensors A and B and output tensor Z and distribute the chunks of those tensors to G PEs. The key point is that after distribution each PE has all the m and n values for the a part of the multiplication but only a part of the k values, so the results of the PE-local computations must be combined together. So the main activity of the algorithm is to do the following:

- Perform the matrix-matrix multiplications for all values of m and n for a subset of the k values. Given the distribution pattern, this corresponds to a totally local matrix multiply.
- Send the result of the local matrix multiply to the partitions of the outputs that need it and reduce.

The trick is doing the reduction efficiently…

Partitioned Attention

Attention - Tensor Parallel

$$I_{b,m,d} = IR_{b,m,c} \times WI_{c,d}$$

$$K_{b,m,e} = I_{b,m,d} \times WK_{d,e}$$

$$Q_{b,m,e} = I_{b,m,d} \times WQ_{d,e}$$

$$QK_{b,m,p}^{B,M,P=M} = Q_{b,p,e}^{B,M,E} \times K_{b,m,e}$$

$$SN_{b,m,p} = exp(QK_{b,m,p})$$

$$SD_{b,p} = SN_{b,m,p}$$

$$A_{b,m,p} = SN_{b,m,p} / SD_{b,p}$$

$$V_{b,m,f} = I_{b,m,d} \times WV_{d,f}$$

$$AV_{b,p,f}^{B,P=M,F} = A_{b,m,p} \times V_{b,m,f}$$

$$Z_{b,p,g} = AV_{b,p,f} \times WZ_{f,g}$$

$$I_{b1,b0,m,d} = IR_{b1,b0,m,c} \times WI_{c,d}$$

$$K_{b1,b0,m,e} = I_{b1,b0,m,d} \times WK_{d,e}$$

$$Q_{b1,b0,m,e} = I_{b1,b0,m,d} \times WQ_{d,e}$$

$$QK_{b1,b0,m,p}^{B1,B0,,M,P=M} = Q_{b1,b0,p,e}^{B1,B0,,M,E} \times K_{b1,b0,m,e}$$

$$SN_{b1,b0,m,p} = exp(QK_{b1,b0,m,p})$$

$$SD_{b1,b0,p} = SN_{b1,b0,m,p}$$

$$A_{b1,b0,m,p} = SN_{b1,b0,m,p} / SD_{b1,b0,p}$$

$$V_{b1,b0,m,f} = I_{b1,b0,m,d} \times WV_{d,f}$$

$$AV_{b1,b0,p,f}^{B1,B0,,P=M,F} = A_{b1,b0,m,p} \times V_{b1,b0,m,f}$$

$$Z_{b1,b0,p,g} = AV_{b1,b0,p,f} \times WZ_{f,g}$$

Run $B1$ in parallel

Attention - Head Parallel

$$K_{b,h,m,e} = I_{b,m,d} \times WK_{d,h,e}$$

$$Q_{b,h,m,e} = I_{b,m,d} \times WQ_{d,h,e}$$

$$QK^{B,H,M,P=M}_{b,h,m,p} = Q^{B,H,M,E}_{b,h,p,e} \times K_{b,h,m,e}$$

$$SN_{b,h,m,p} = exp(QK_{b,h,m,p})$$

$$SD_{b,h,p} = SN_{b,h,m,p}$$

$$A_{b,h,m,p} = SN_{b,h,m,p} / SD_{b,h,p}$$

$$V_{b,h,m,f} = I_{b,m,d} \times WV_{d,h,f}$$

$$AV^{B,H,P=M,F}_{b,h,p,f} = A_{b,h,m,p} \times V_{b,h,m,f}$$

$$C^{B,P=M,G=H\times F}_{b,p,h\times F+f} = AV_{b,h,p,f}$$

$$Z_{b,p,d} = C_{b,p,f} \times WZ_{g,d}$$

$$K_{b,h1,h0,m,e} = I_{b,m,d} \times WK_{d,h1,h0,e}$$

$$Q_{b,h1,h0,m,e} = I_{b,m,d} \times WQ_{d,h1,h0,e}$$

$$QK^{B,H1,H0,M,P=M}_{b,h1,h0,m,p} = Q^{B,H1,H0,M,E}_{b,h1,h0,p,e} \times K_{b,h1,h0,m,e}$$

$$SN_{b,h1,h0,m,p} = exp(QK_{b,h1,h0,m,p})$$

$$SD_{b,h1,h0,p} = SN_{b,h1,h0,m,p}$$

$$A_{b,h1,h0,m,p} = SN_{b,h1,h0,m,p} / SD_{b,h1,h0,p}$$

$$V_{b,h1,h0,m,f} = I_{b,m,d} \times WV_{d,h1,h0,f}$$

$$AV^{B,H1,H0,P=M,F}_{b,h1,h0,p,f} = A_{b,h1,h0,m,p} \times V_{b,h1,h0,m,f}$$

$$C^{B,P=M,G=H1\times H0\times F}_{b,p,(h1\times H0+h0)\times F+f} = AV_{b,h1,h0,p,f}$$

$$Z_{b,p,d} = C_{b,p,f} \times WZ_{g,d}$$

Run $H1$ in parallel

Attention - Data Parallel

$$K_{b,h,m,e} = I_{b,m,d} \times WK_{d,h,e}$$

$$Q_{b,h,m,e} = I_{b,m,d} \times WQ_{d,h,e}$$

$$QK^{B,H,M,P=M}_{b,h,m,p} = Q^{B,H,M,E}_{b,h,p,e} \times K_{b,h,m,e}$$

$$SN_{b,h,m,p} = exp(QK_{b,h,m,p})$$

$$SD_{b,h,p} = SN_{b,h,m,p}$$

$$A_{b,h,m,p} = SN_{b,h,m,p} / SD_{b,h,p}$$

$$V_{b,h,m,f} = I_{b,m,d} \times WV_{d,h,f}$$

$$AV^{B,H,P=M,F}_{b,h,p,f} = A_{b,h,m,p} \times V_{b,h,m,f}$$

$$C^{B,P=M,G=H\times F}_{b,p,h\times F+f} = AV_{b,h,p,f}$$

$$Z_{b,p,d} = C_{b,p,f} \times WZ_{g,d}$$

$$K_{b,h1,h0,m,e} = I_{b,m,d1,d0} \times WK_{d1,d0,h1,h0,e}$$

$$Q_{b,h1,h0,m,e} = I_{b,m,d1,d0} \times WQ_{d1,d0,h1,h0,e}$$

$$QK^{B,H1,H0,M,P=M}_{b,h1,h0,m,p} = Q^{B,H1,H0,M,E}_{b,h1,h0,p,e} \times K_{b,h1,h0,m,e}$$

$$SN_{b,h1,h0,m,p} = exp(QK_{b,h1,h0,m,p})$$

$$SD_{b,h1,h0,p} = SN_{b,h1,h0,m,p}$$

$$A_{b,h1,h0,m,p} = SN_{b,h1,h0,m,p} / SD_{b,h1,h0,p}$$

$$V_{b,h1,h0,m,f} = I_{b,m,d1,d0} \times WV_{d1,d0,h1,h0,f}$$

$$AV^{B,H1,H0,P=M,F}_{b,h1,h0,p,f} = A_{b,h1,h0,m,p} \times V_{b,h1,h0,m,f}$$

$$C^{B,P=M,G=H1\times H0\times F}_{b,p,(h1\times H0+h0)\times F+f} = AV_{b,h1,h0,p,f}$$

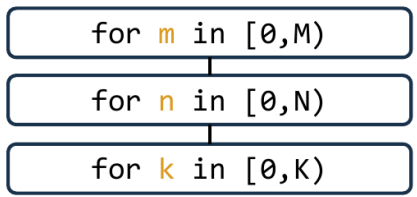
$$Z_{b,p,d1,d0} = C_{b,p,f} \times WZ_{g,d1,d0}$$

Run $H1$ and $D1$ in parallel

Looptree – Dataflow and Dataplacement

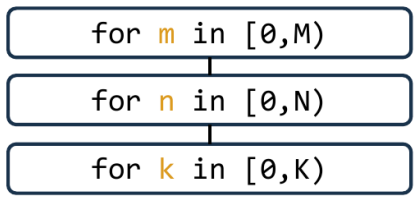
Looptree - Dataflow

$$Z_{m,n} = A_{m,k} \times B_{n,k}$$



Looptree - Dataflow

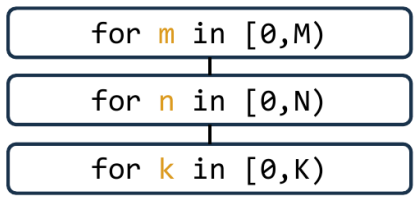
$$Z_{m,n} = A_{m,k} \times B_{n,k}$$



What is stationary?

Looptree - Dataflow

$$Z_{m,n} = A_{m,k} \times B_{n,k}$$

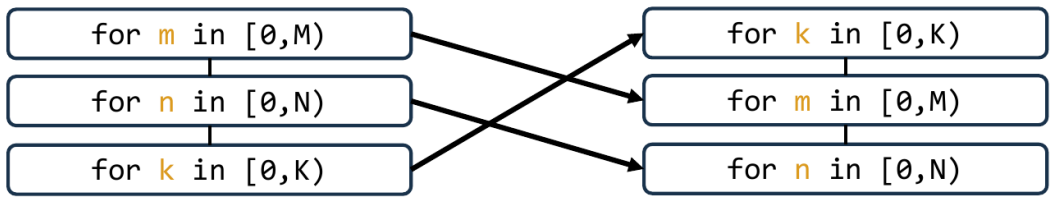


What is stationary?

Output elements

Looptree - Dataflow

$$Z_{m,n} = A_{m,k} \times B_{n,k}$$

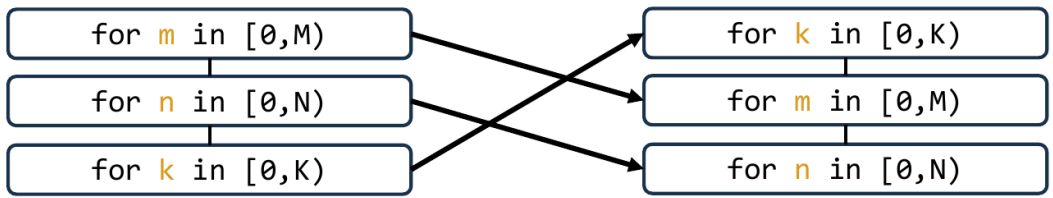


What is stationary?

Output elements

Looptree - Dataflow

$$Z_{m,n} = A_{m,k} \times B_{n,k}$$



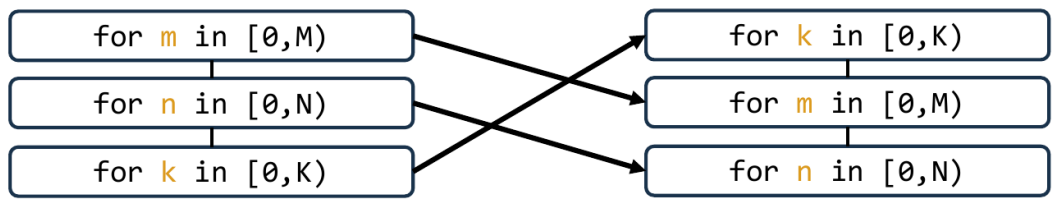
What is stationary?

Output elements

What is stationary?

Looptree - Dataflow

$$Z_{m,n} = A_{m,k} \times B_{n,k}$$



What is stationary?

Output elements

What is stationary?

Elements of A

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

```
for m1 in [0,M1)
for n1 in [0,N1)
for k1 in [0,K1)
for m0 in [0,M0)
for n0 in [0,N0)
for k0 in [0,K0)
```

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

```
for m1 in [0,M1)
for n1 in [0,N1)
for k1 in [0,K1)
for m0 in [0,M0)
for n0 in [0,N0)
for k0 in [0,K0)
```

What is stationary?

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

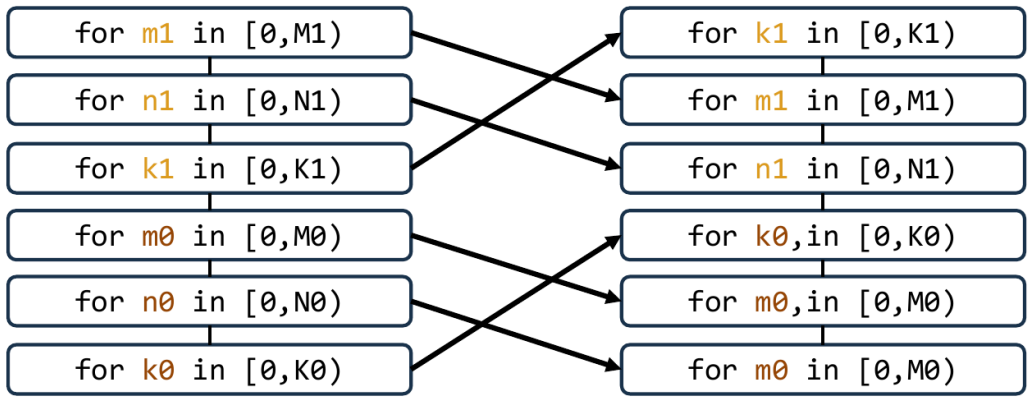
```
for m1 in [0,M1)
for n1 in [0,N1)
for k1 in [0,K1)
for m0 in [0,M0)
for n0 in [0,N0)
for k0 in [0,K0)
```

What is stationary?

Output tiles and output elements

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

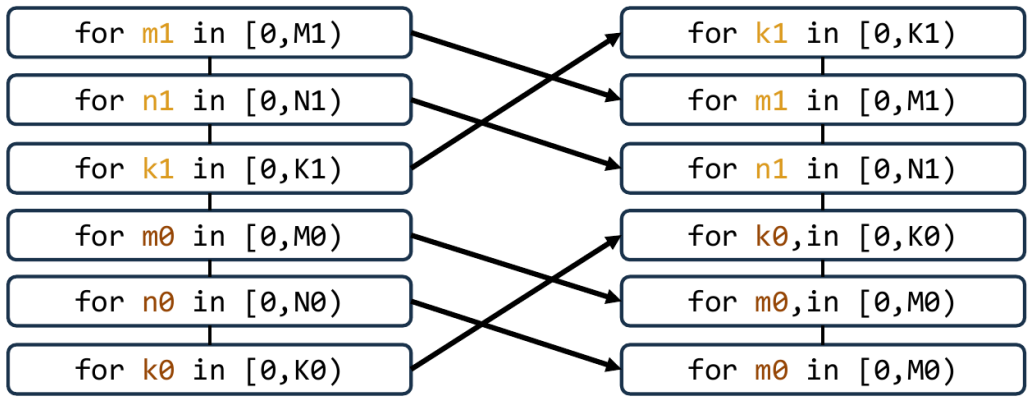


What is stationary?

Output tiles and output elements

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$



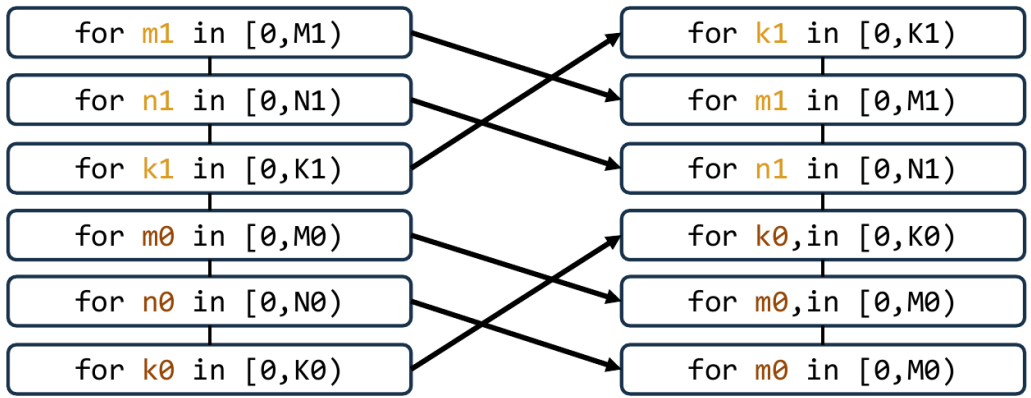
What is stationary?

What is stationary?

Output tiles and output elements

Looptree – Partitioned Dataflow

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$



What is stationary?

Output tiles and output elements

What is stationary?

A tiles and A elements

Looptree - Dataflow and Dataplacement

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

Dataflow

```
for m1 in [0,M1)
for n1 in [0,N1)
for k1 in [0,K1)
for m0 in [0,M0)
for n0 in [0,N0)
for k0 in [0,K0)
```

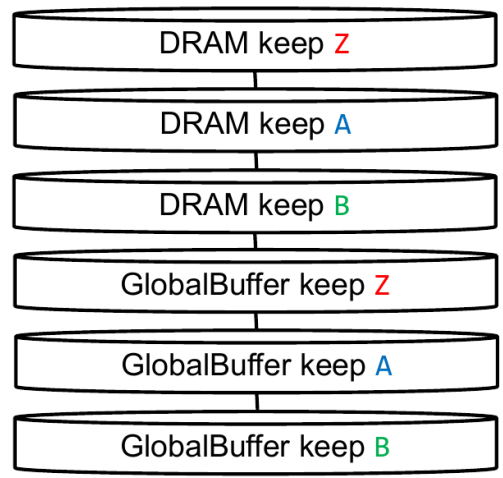
Looptree - Dataflow and Dataplacement

$$Z_{m1,\mathrm{m0},n1,\mathrm{n0}} = A_{m1,\mathrm{m0},k1,\mathrm{k0}} \times B_{n1,\mathrm{n0},k1,\mathrm{k0}}$$

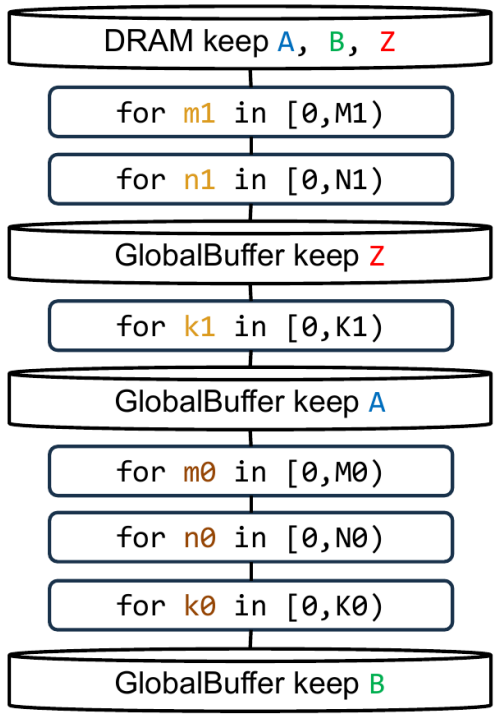
Dataflow

```
for m1 in [0,M1)
for n1 in [0,N1)
for k1 in [0,K1)
for m0 in [0,M0)
for n0 in [0,N0)
for k0 in [0,K0)
```

Dataplacement

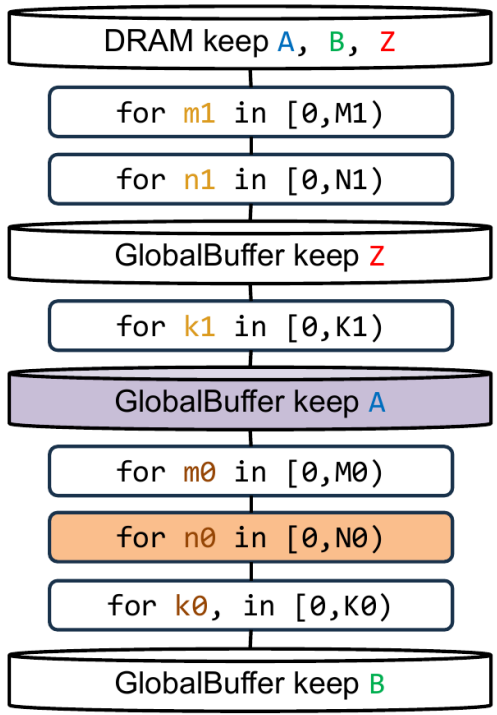


Looptree - Dataflow and Dataplacement



$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

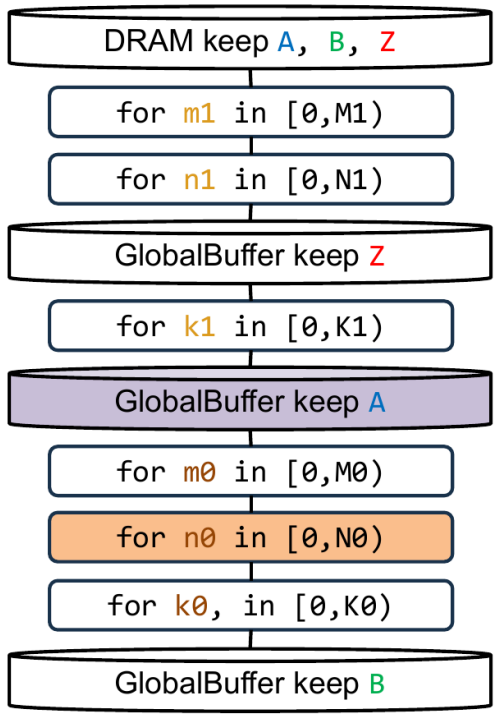
Looptree – Below/Not In



$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Looptree – Below/Not In

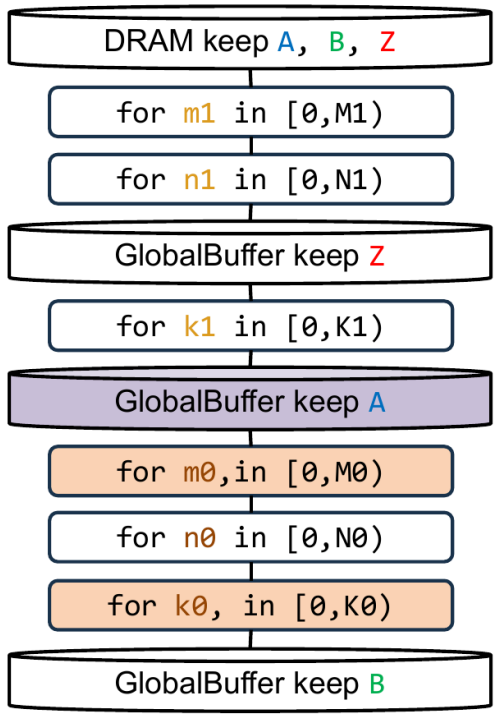


$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Number of times each element of A is reused

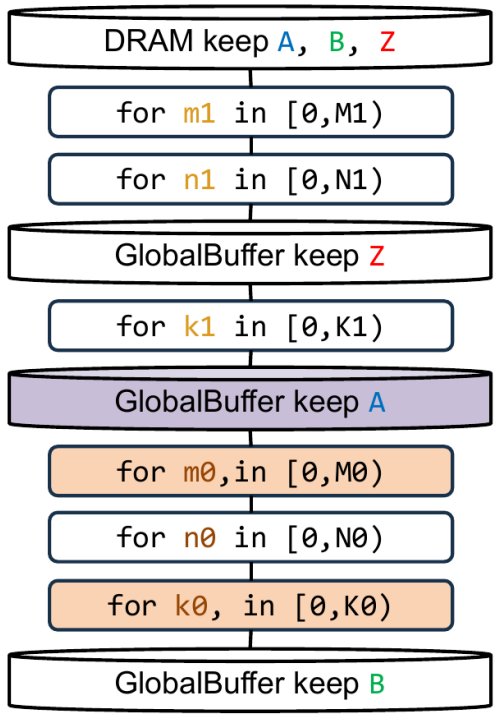
Looptree – Below/In



$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Looptree – Below/In

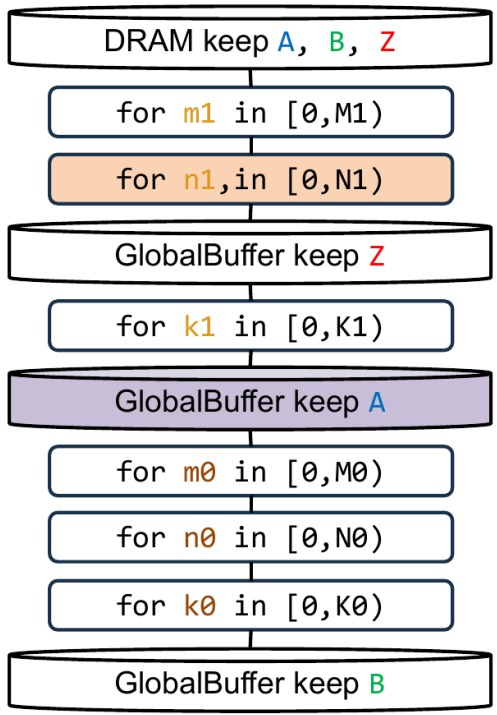


$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Shape of A's data in the buffer

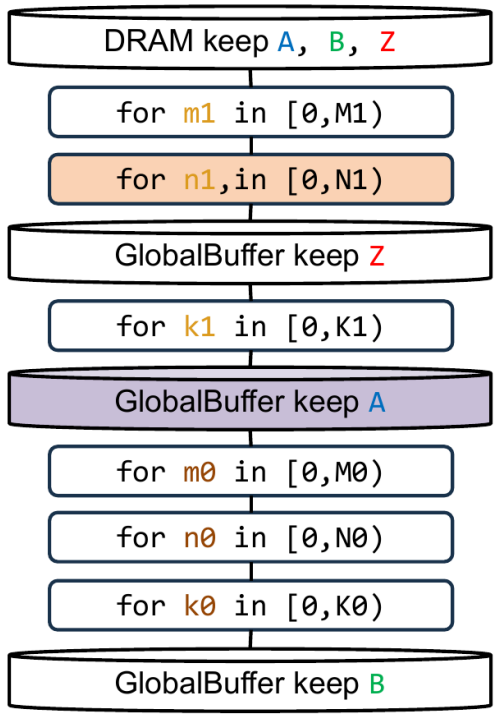
Looptree – Above/Not In



$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Looptree – Above/Not In

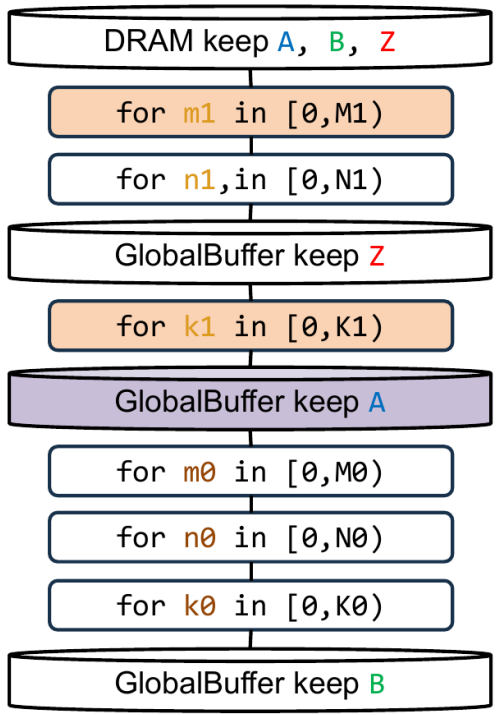


$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Number of times each chunk of A's data is fetched into the buffer

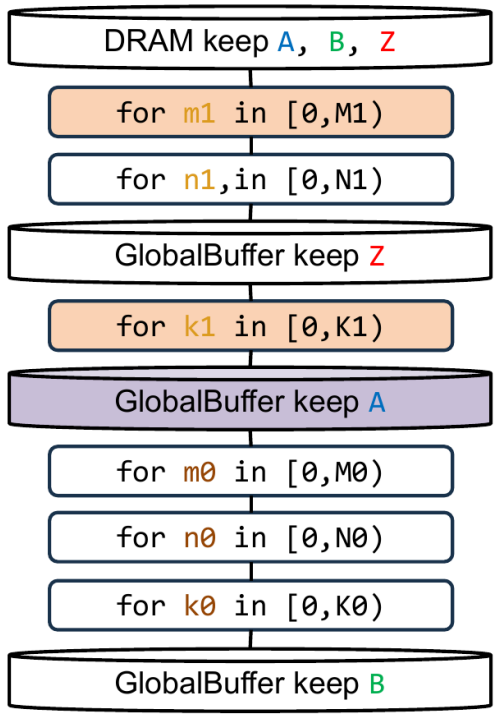
Looptree – Above/In



$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Looptree – Above/In

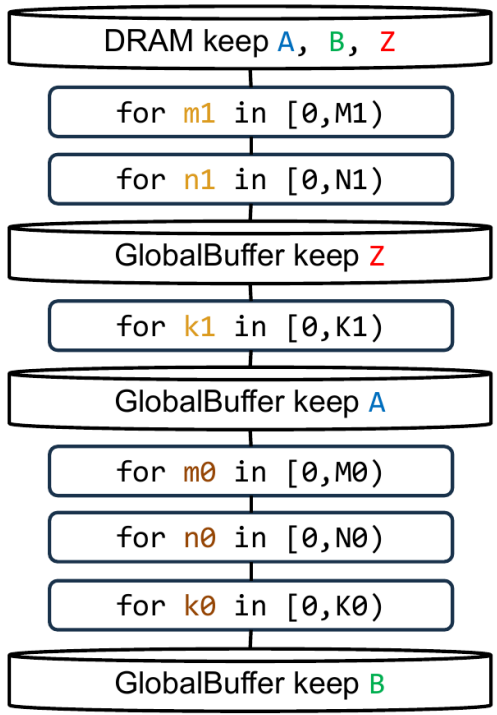


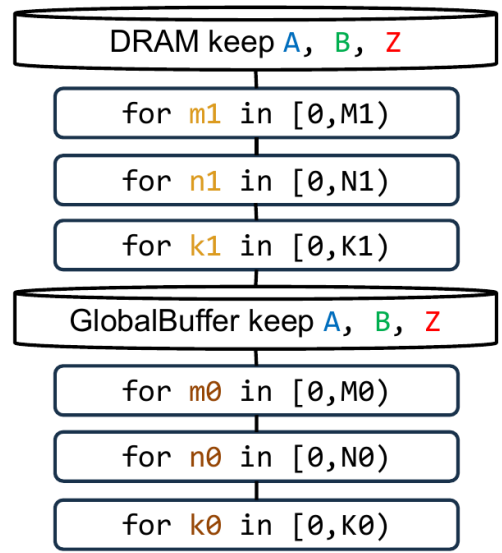
$$Z_{m1,m0,n1,n0} = A_{m1,m0,k1,k0} \times B_{n1,n0,k1,k0}$$

	Loop Above Storage Node	Loop Below Storage Node
Rank not in stored tensor		
Rank is in Stored Tensor		

Number of distinct chunks of A's data that is fetched into the buffer

Customary Dataflow and Dataplacement





Most prior art kept all tensors in same spot

Mapping Process

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

- A dataflow (**for** loop order)

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

- A dataflow (**for** loop order)
- A dataplacement (including bypassing)

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

- A dataflow (**for** loop order)
- A dataplacement (including bypassing)
- A compute placement (e.g., parallelization)

Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

- A dataflow (**for** loop order)
- A dataplacement (including bypassing)
- A compute placement (e.g., parallelization)
- Partition sizing

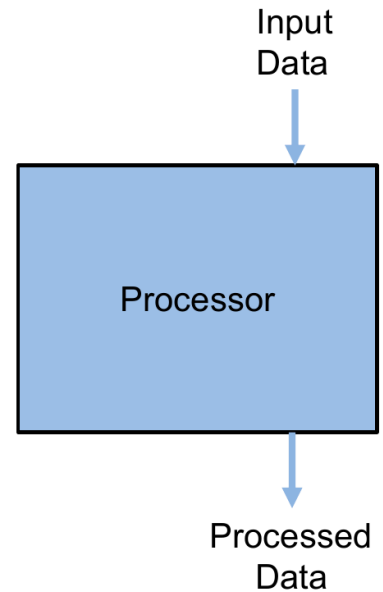
Mapping

Definition: selecting the placement and scheduling in space and time of every operation (including delivering the appropriate operands) required for a computation onto the hardware function units of the accelerator.

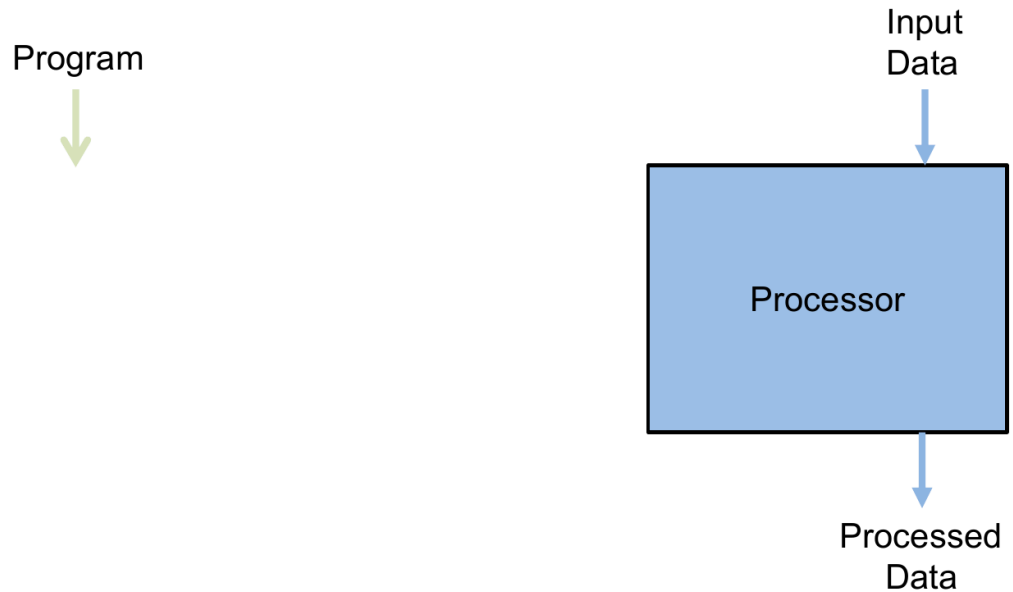
Steps: Within the constraints of the hardware, select for each level of the storage hierarchy:

- A dataflow (**for** loop order)
- A dataplacement (including bypassing)
- A compute placement (e.g., parallelization)
- Partition sizing
- A binding computation to specific hardware units

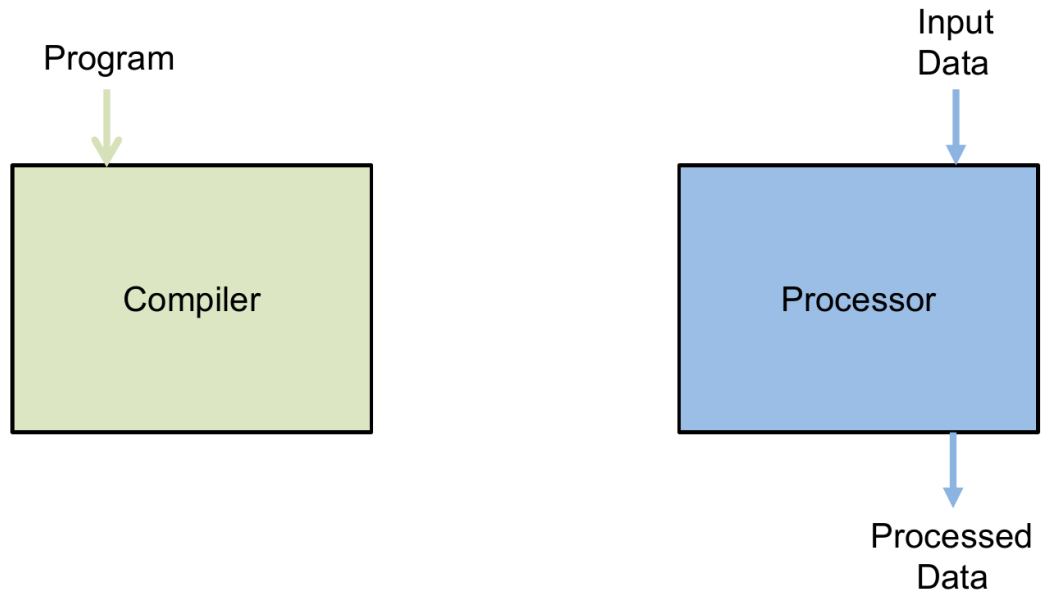
CPU Compute Model



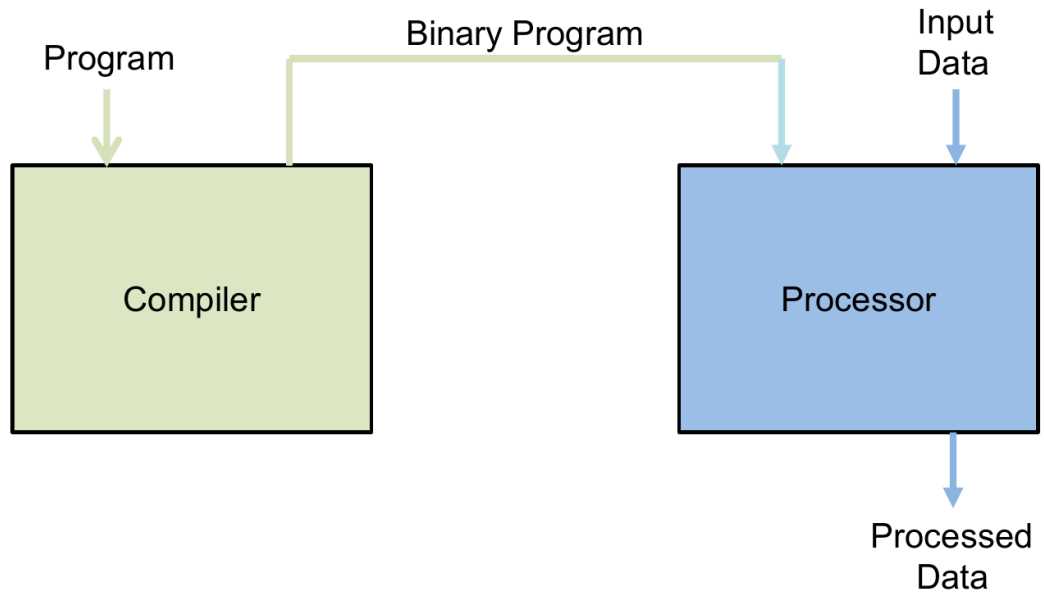
CPU Compute Model



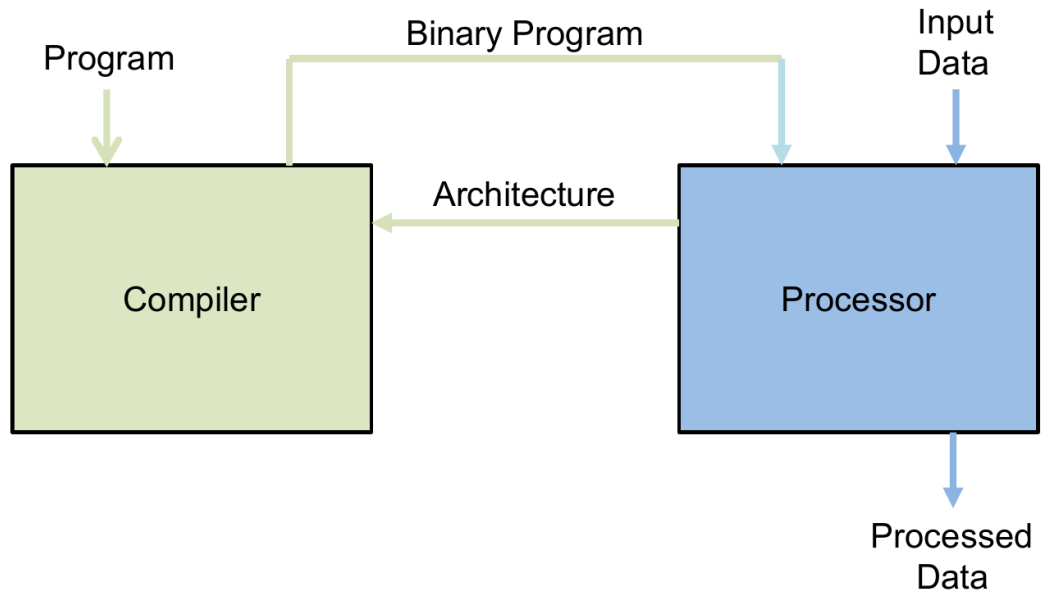
CPU Compute Model



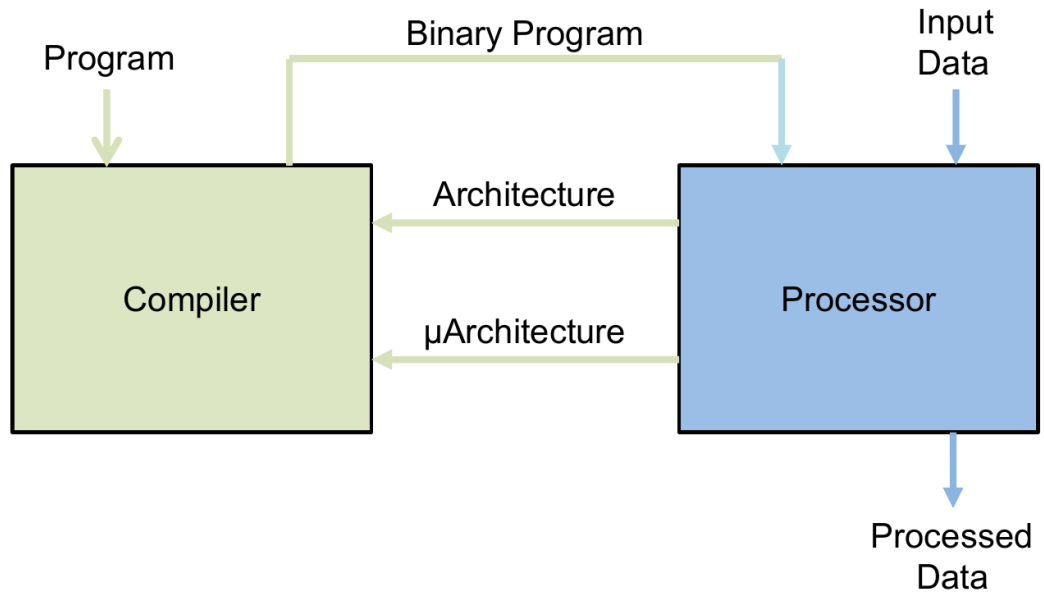
CPU Compute Model



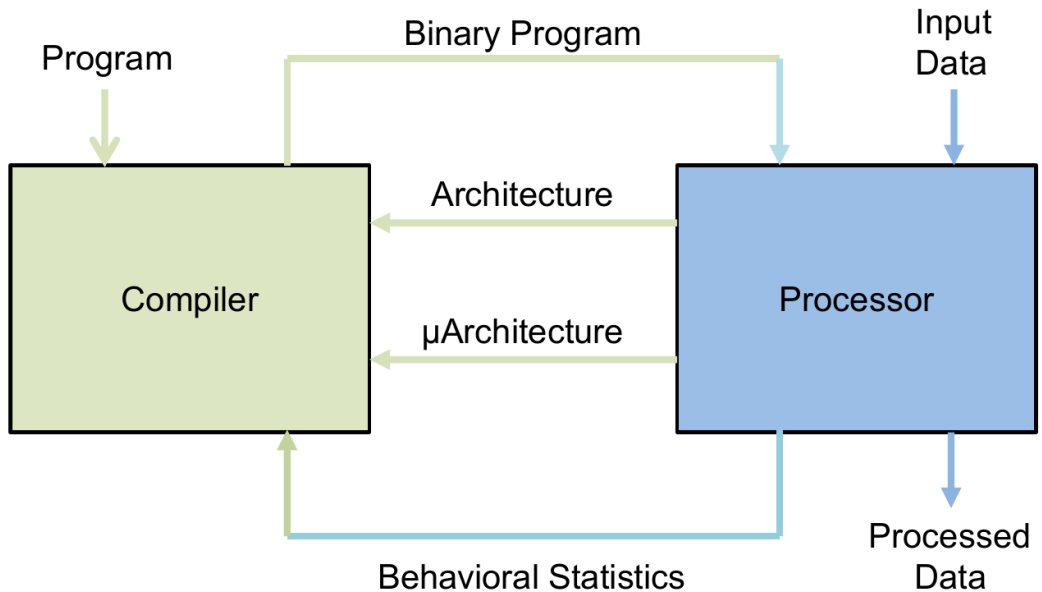
CPU Compute Model



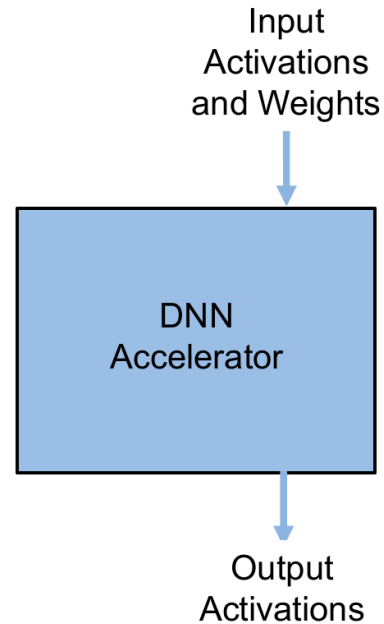
CPU Compute Model



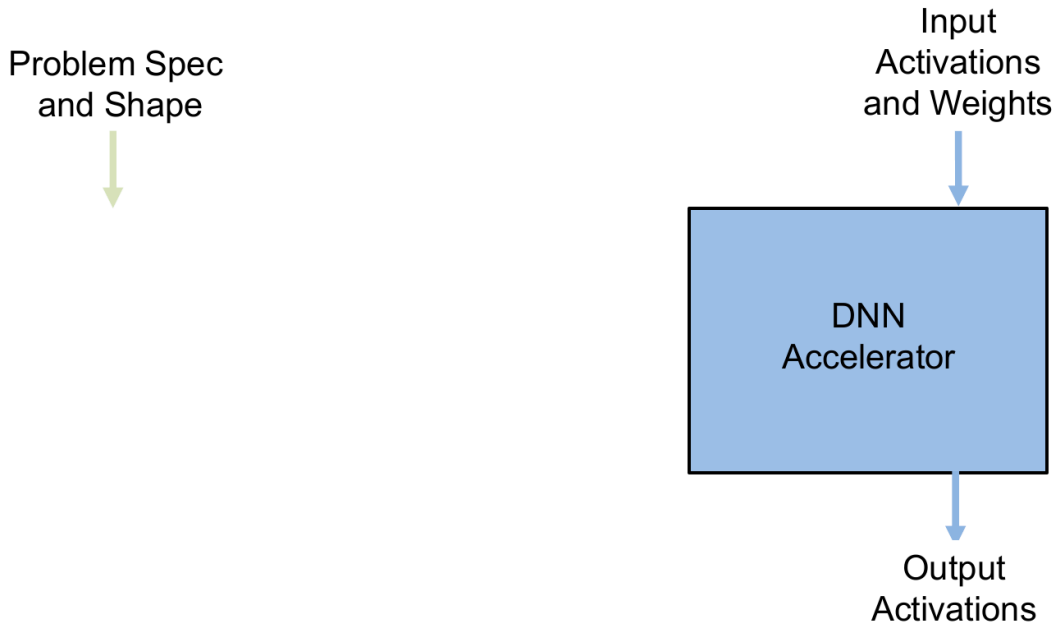
CPU Compute Model



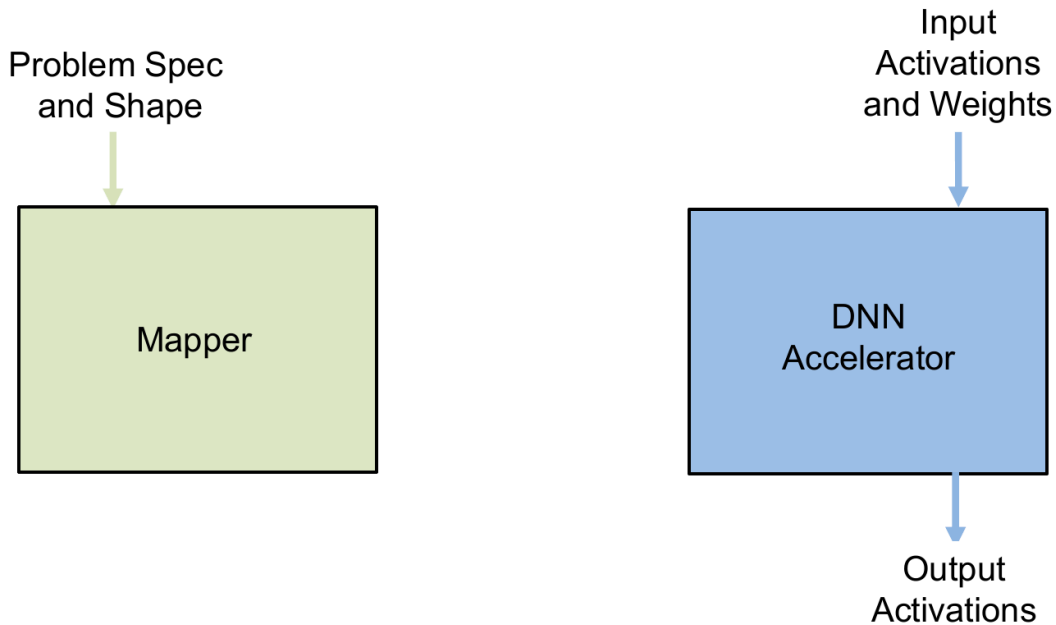
DNN Compute Model



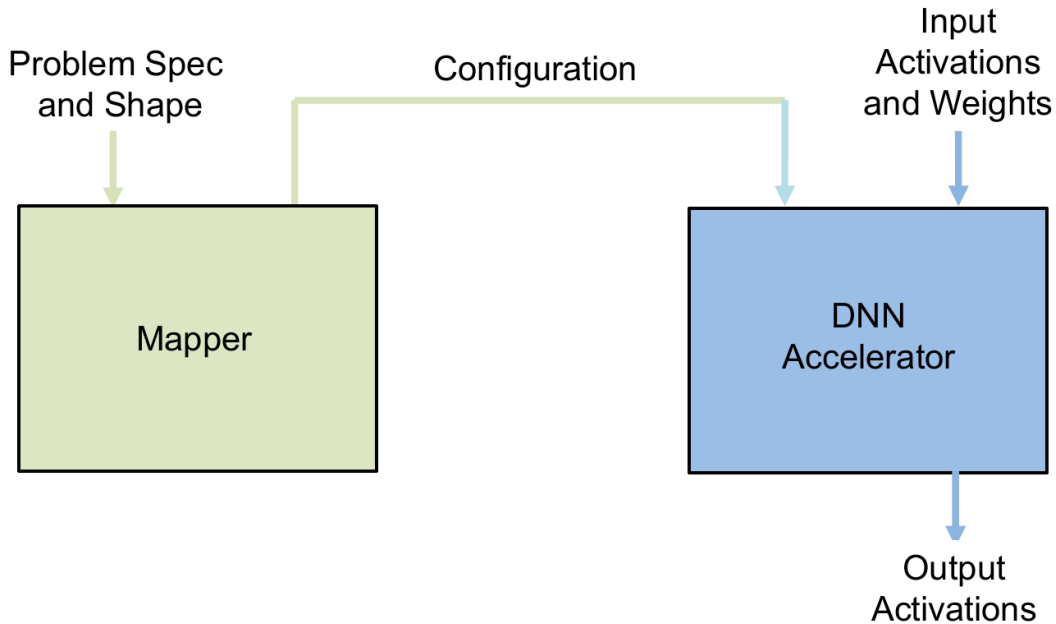
DNN Compute Model



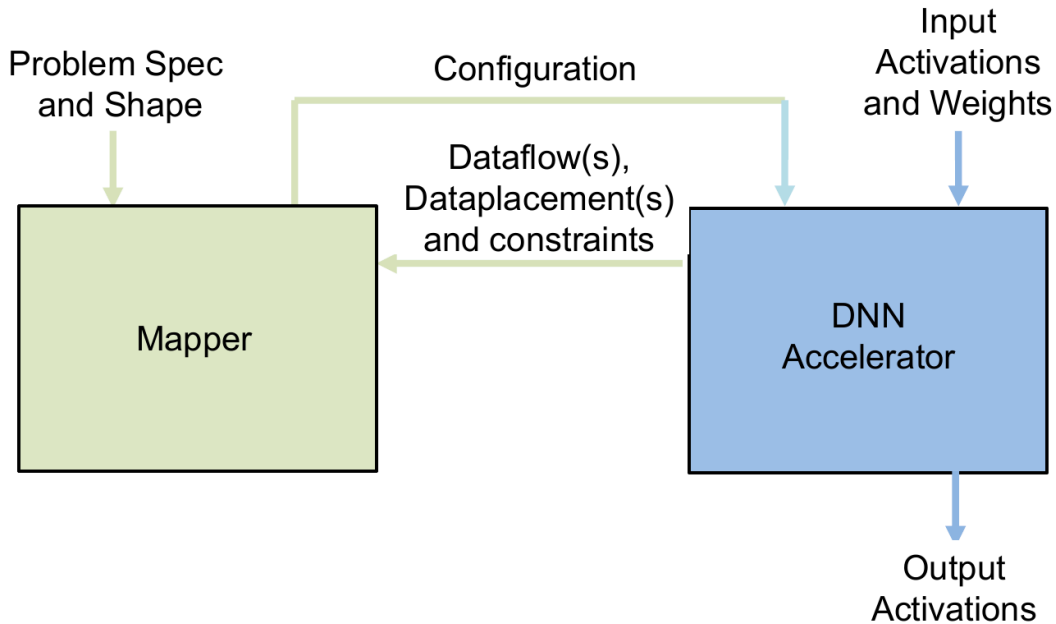
DNN Compute Model



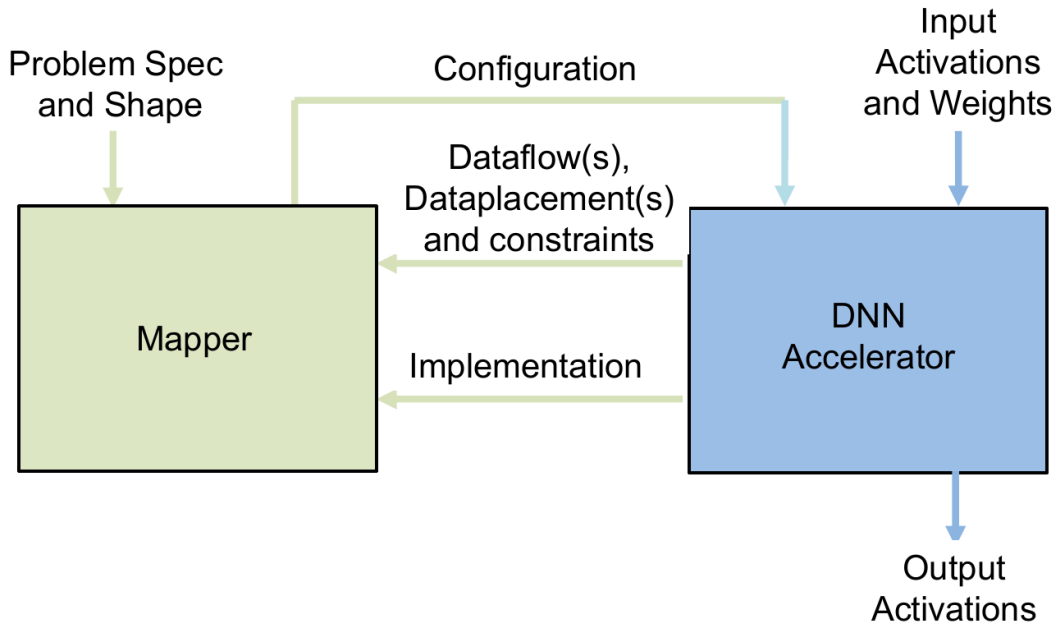
DNN Compute Model



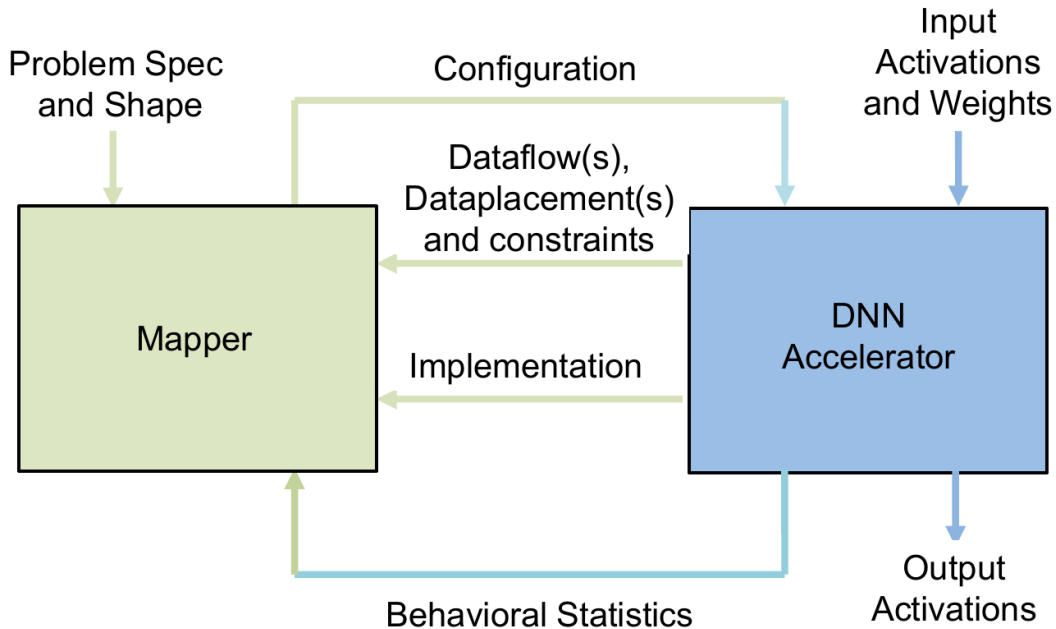
DNN Compute Model



DNN Compute Model

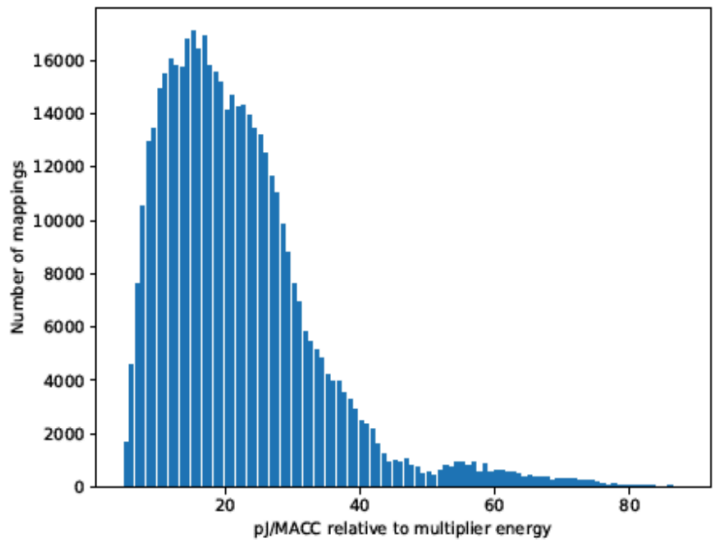


DNN Compute Model



Mapping Choices

Energy-efficiency of peak-perf mappings of a single problem



480,000 mappings shown

Spread: 19x in energy efficiency

Only 1 is optimal, 9 others within 1%

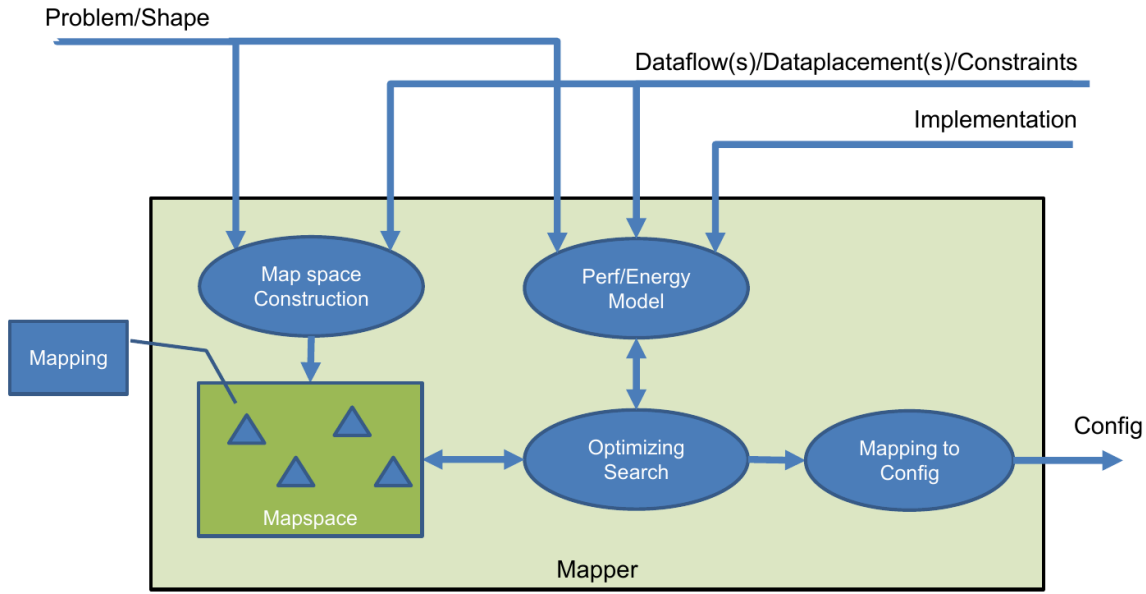
A model needs a mapper to evaluate a DNN workload on an architecture

6,582 mappings have min. DRAM accesses but vary 11x in energy efficiency

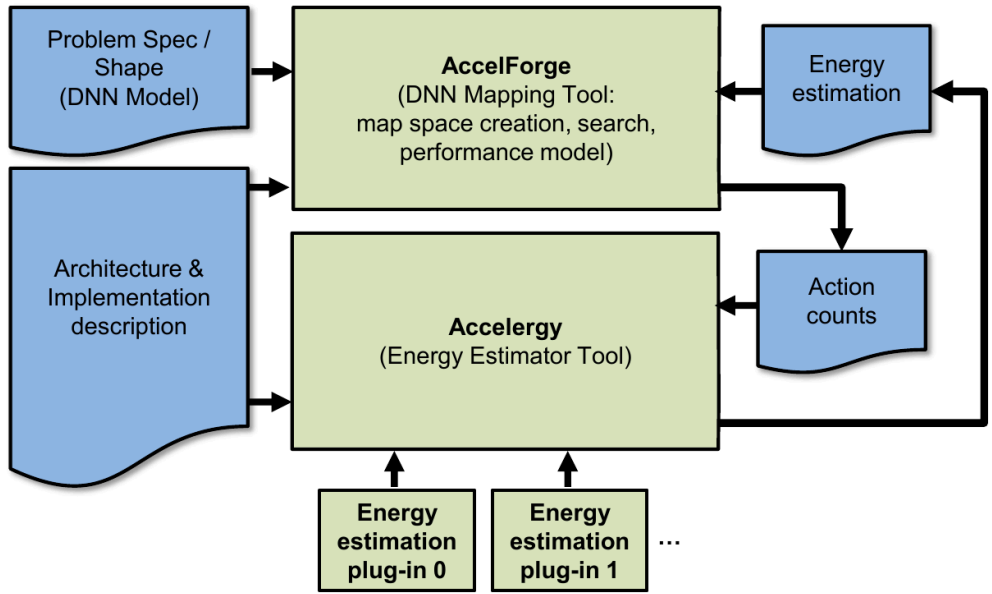
A mapper needs a good cost model to find an optimal mapping

Source: Parashar, Timeloop

Mapper Organization



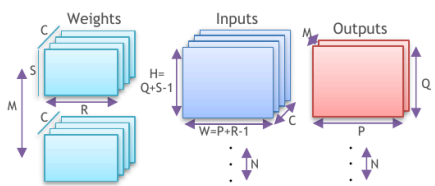
AccelForge/Accelergy



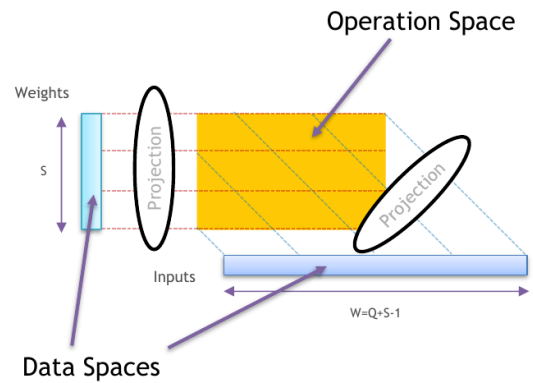
Workload Specification

Provided to AccelForge

$$O_{n,m,p,q} = I_{n,c,Up+r,Uq+s} \times F_{m,c,r,s}$$



Inferred by Accelforge



Determine storage requirements)

Source: Parashar, Timeloop

Architecture Specifications

- Temporal reuse features
  - Number of buffer levels and buffer sizes
  - Buffer bypassing capabilities
  - Network topology
  - …
- Parallelism and spatial reuse features
  - Topology of spatial fractures
  - Multicast capabilities
  - Inter-PE network, e.g., spatial sum and forwarding reuse
  - …
- Constraints
  - Index sequence restrictions, e.g., allowable strides
  - Fixed level 0 loop nest, e.g., fixed vector width
  - Fixed level 1 spatial mappings, e.g., input/output channel array
  - …

Architecture determines the legal mappings:
loop permutations (dataflows) and associated loop limits

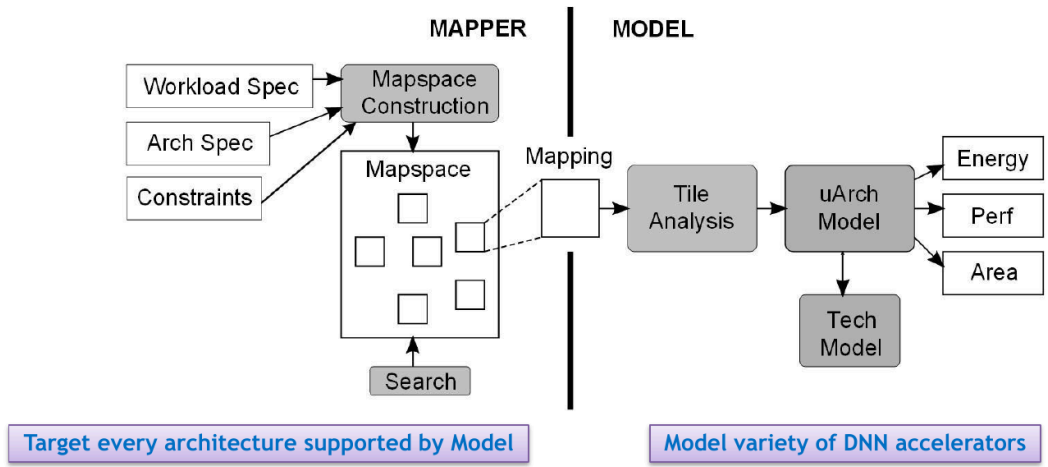
Implementation Specifications

- Buffer bandwidth and latency
- Buffer port and banking organization
- PE vectorization
- Network bandwidth and latency, e.g., router costs
- Shared or per-datatype network links
- …

Implementation determines the latency and energy consumption of a mapping

AccelForge

Tool for Evaluation and Architectural Design-Space Exploration of DNN Accelerators



Source: Parashar, Timeloop

Explore Search algorithm in Final Project!