TCP: A Tensor Contraction Processor for AI Workloads

Weiming Hu

2024.09.05

Outline

TCP: A Tensor Contraction Processor for AI, ISCA'24, Industry Track

- Institude: Furiosa AI, start from 2017

Toolkit

- SDK
- NPU

Other chips

- Meta maia
- Cerebras
- Sambanova

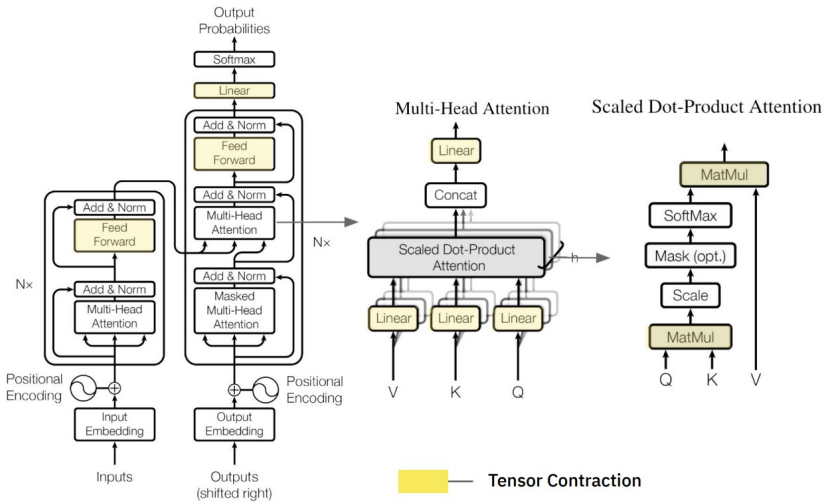
Background

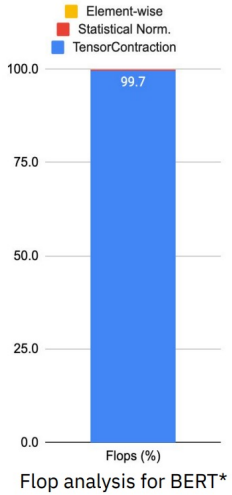
Tensor Contraction

- The fundamental core computation of deep learning is tensor contraction, higher dimensional generalizations of matrix multiplication.

Tensor contraction is the core computation in Deep Learning

- 99.7% FLOP from tensor contraction in BERT





Flop analysis for BERT*

Tensor Contraction

Matrix Multiplication

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

$$C_{ij} = \sum_k A_{ik} B_{kj} = A_{ik} B_{kj}$$

$$C_{ijkl} = \sum_m \sum_n A_{ijmn} B_{kmln}$$

Low-Level Einsum as Primitive

Einstein summation

Low-level einsum

=tensor contraction + **explicit memory layout** + **explicit scheduling**

Torch.einsum provide a abstract API of tensor OP for Einstein summation

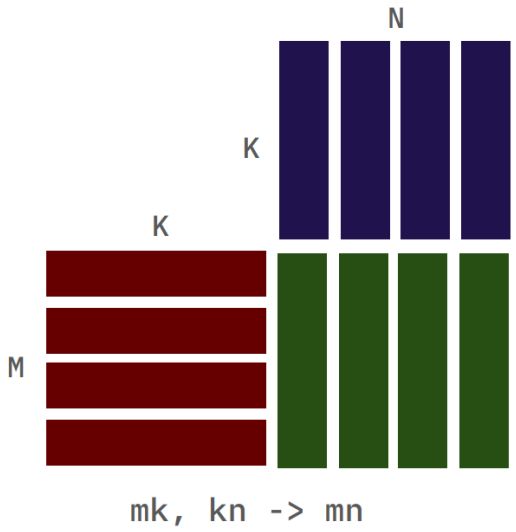
Low-level einsum can be executed efficiently in TCP architecture

$$C_{ijkl} = \sum_{m}\sum_{n} A_{ijmn} B_{kmln}$$

```
torch.einsum('ijmn,kmln->ijkl', [a, b])
```

Einsum notation for tensor contraction

A Whole Tensor Contraction as a Primitive

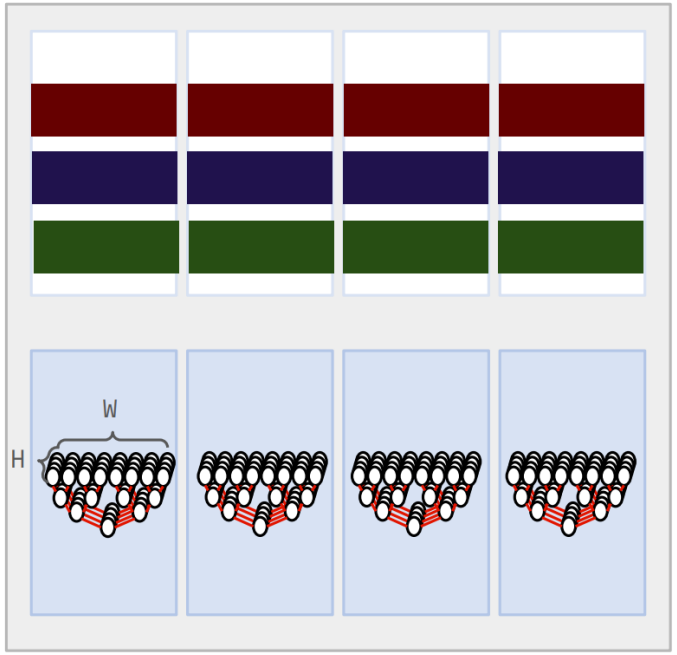



```
// spatial mapping
for (n_blk in 0..4) {

  // temporal scheduling
  for (m_blk in 0..4) {
    for (k_blk in 0..(K/W)) {
      for (m_idx in 0..(M/4)) {
        for (n_idx in 0..(N/4/H)) {

          // unit computation
          for (h in 0..H) {
            for (w in 0..W) {
              m = m_index_of(m_blk, m_idx)
              n = n_index_of(n_blk, n_idx, H)
              k = k_index_of(k_blk, W)
              C[m][n] += A[m][k]xB[k][n]

} } } } } } }
```



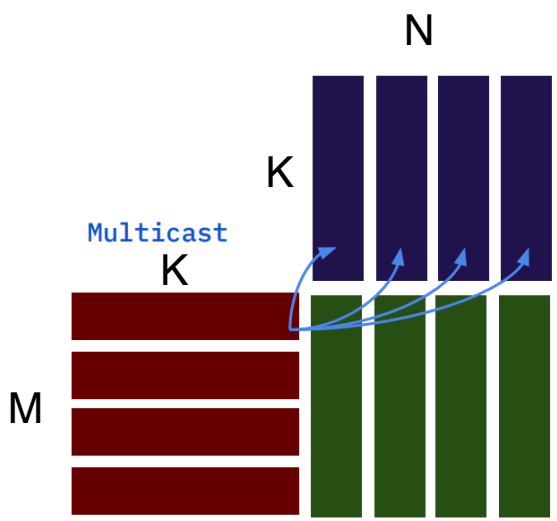
Lowered shape
memory layout of tensors

Tactic
scheduling of computation

a single primitive

Chip

A Whole Tensor Contraction as a Primitive



parallel

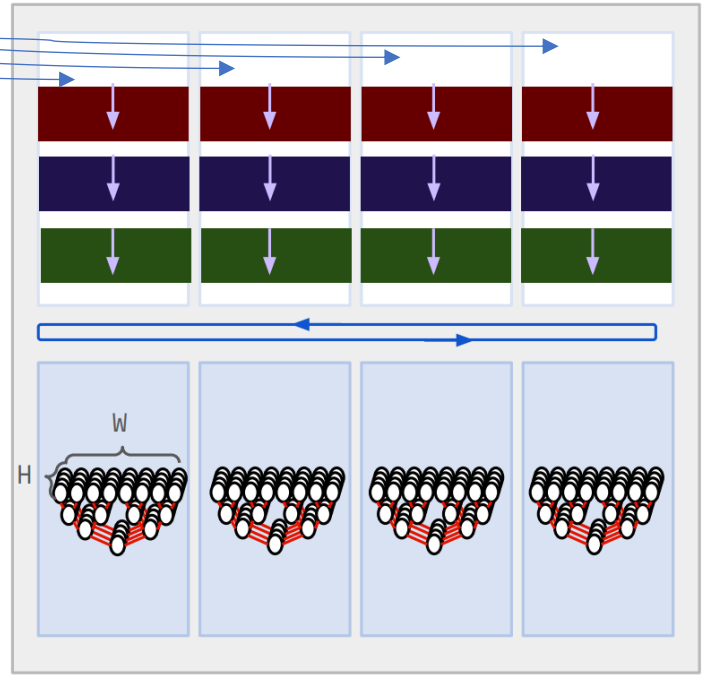
```
// spatial mapping
for (n_blk in 0..4) {

  // temporal scheduling
  for (m_blk in 0..4) {
    for (k_blk in 0..(K/W)) {
      for (m_idx in 0..(M/4)) {
        for (n_idx in 0..(N/4/H)) {

          // unit computation
          for (h in 0..H) {
            for (w in 0..W) {
              m = m_index_of(m_blk, m_idx)
              n = n_index_of(n_blk, n_idx, H)
              k = k_index_of(k_blk, W)
              C[m][n] += A[m][k]xB[k][n]

} } } } } } }
```

Data reuse
K split



temporal access

SRAM

Multicast

Compute Units

Lowered shape
memory layout of tensors

Tactic
scheduling of computation

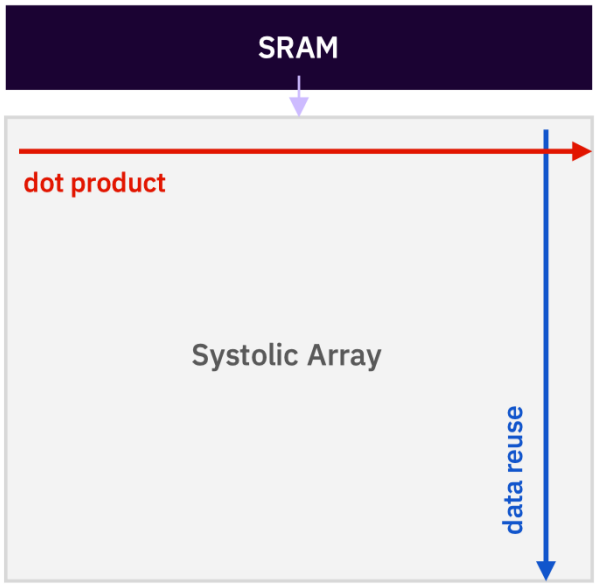
a single primitive

Chip

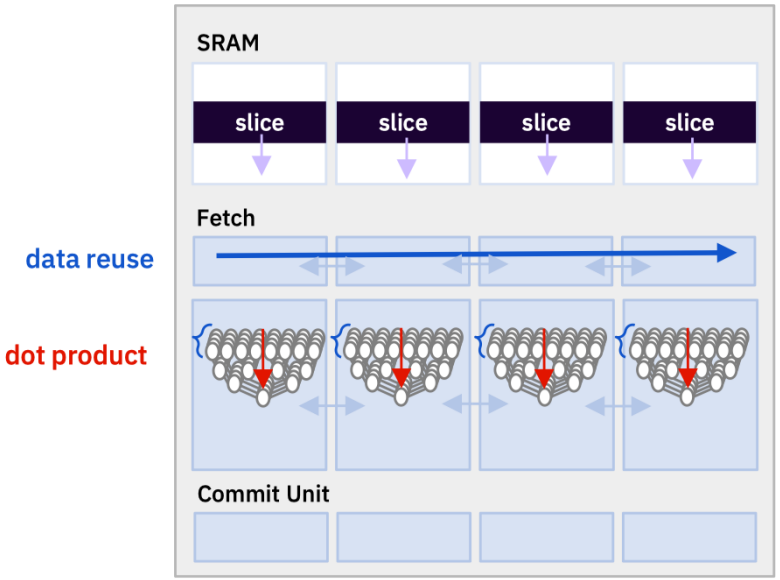
Flexible Reconfigurability

Hardware operates according to software-defined (optimizes) tactics

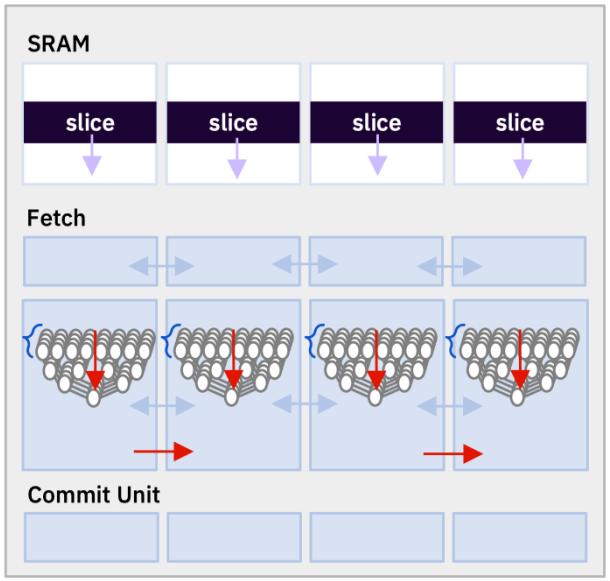
- Data read once from SRAM can be multicast and fed multiple times
- Temporal pipelining allows full utilization of spatially parallel compute units
- All data paths can be streamlined



TPU 128 x 128



TCP W x 4H



TCP (2W x H) x 2

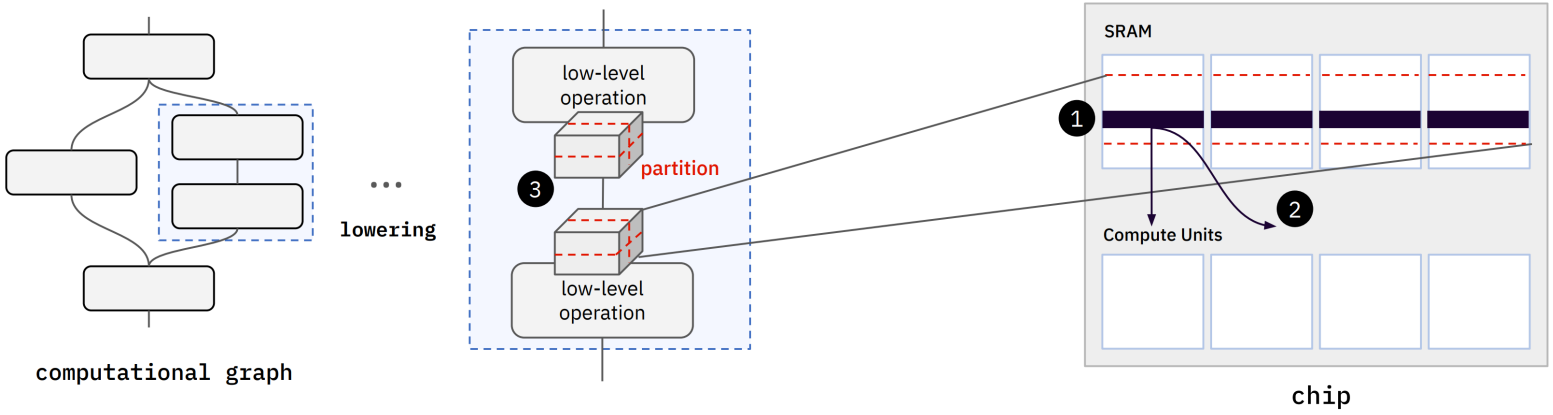
Compiler Lowers Shape for Low-Level Einsum

Mapping tensors onto on-chip memory impacts performance & energy efficiency

- SRAM access performance (e.g., row/column major)
- Data movement of input tensors to compute units for a single low-level operation
- Data movement across operators

Compiler link:

- https://github.com/furiosa-ai/furiosa-sdk
- https://furiosa-ai.github.io/docs/latest/en/

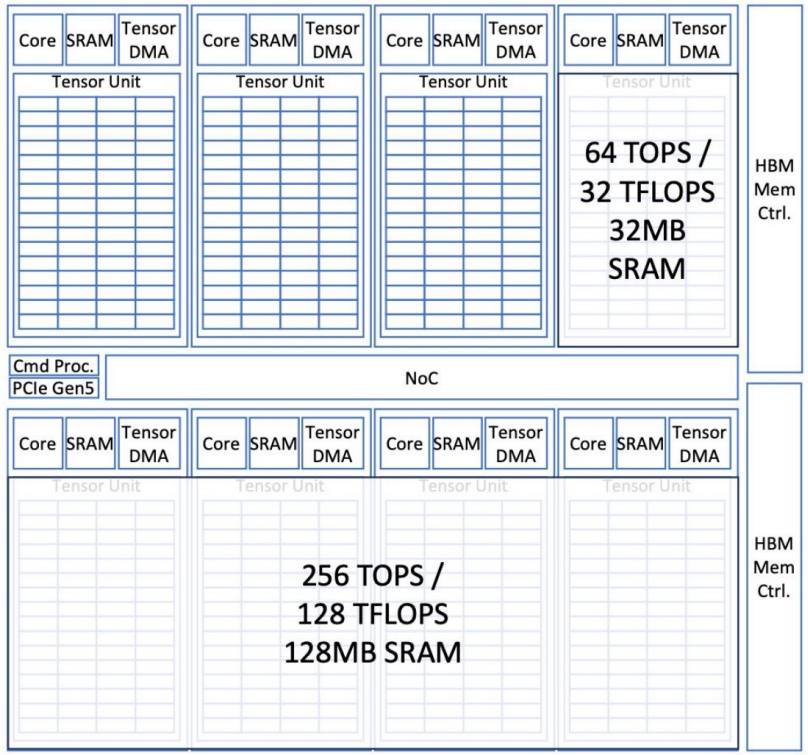


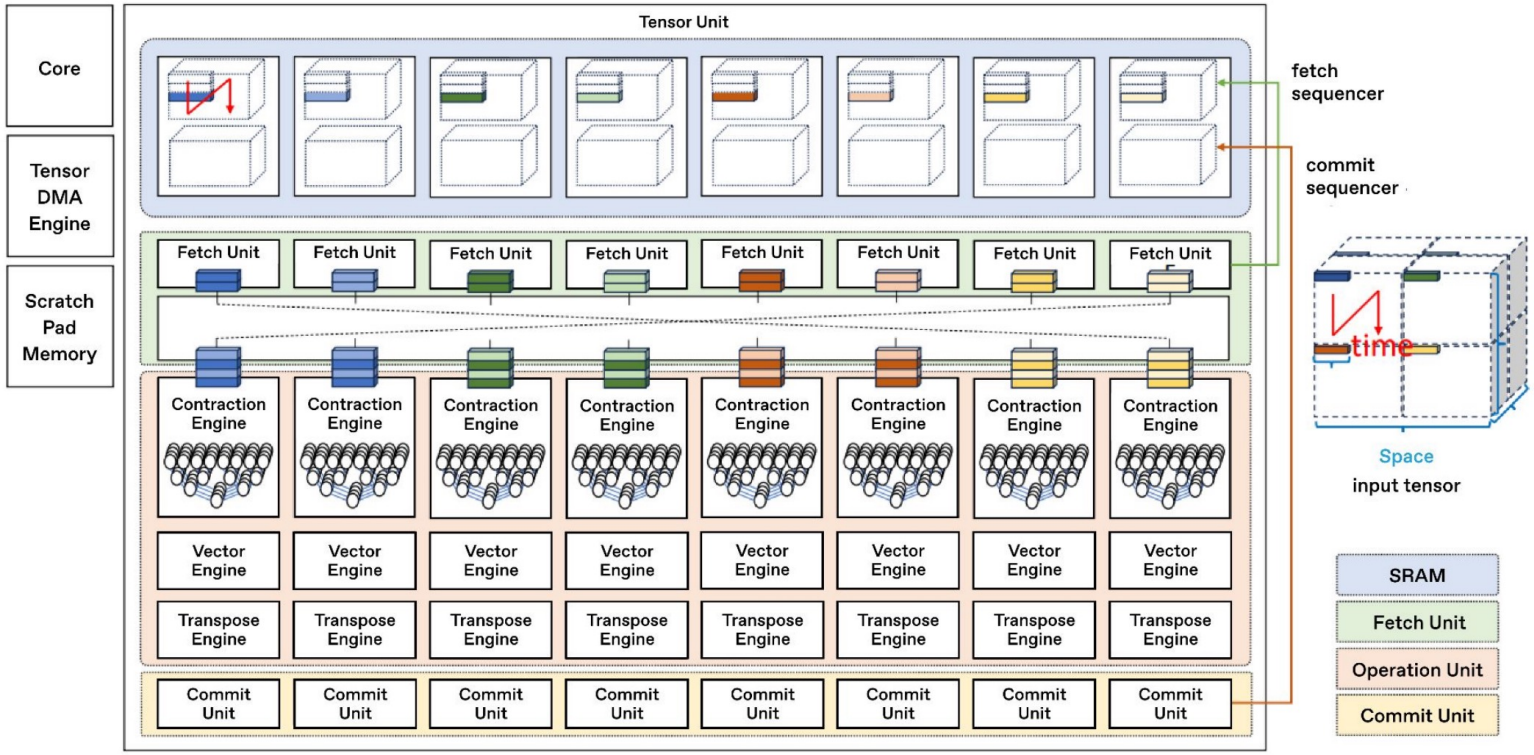
SoC Overview

8 tensor units

Core: scalar processor, control flow

Tensor DMA Engine: manage the data load from DRAM to SRAM



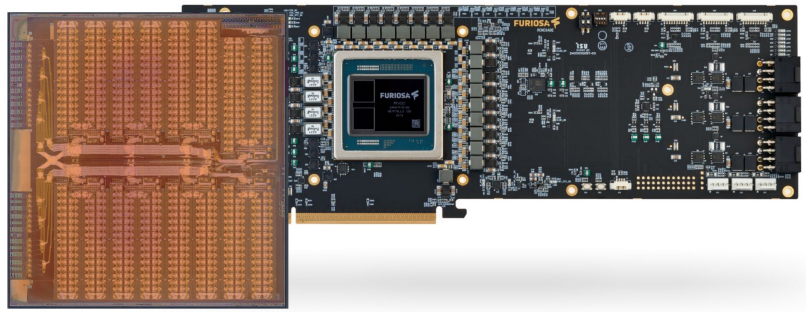


Evaluation

Results from a cycle-level simulator

Advanced process (5nm)

bandwidth and computing power basically match H100 (1/2 bandwidth and ¼ computer power)



512 TFLOPS
64 TFLOPS (FP8) x 8 Processing Elements

1.5 TB/s
Memory Bandwidth

INT8 (512 TOPS), BF16 (256 TFLOPS), INT4 (1 POPS), FP8 (512 TFLOPS)

48 GB
Memory Capacity

150 W TDP
targeting air-cooled datacenters

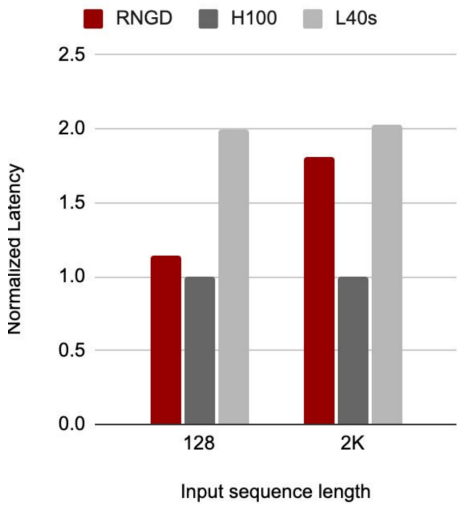
PCIe P2P support For LLMs

256 MB SRAM
384 TB/s On-chip Bandwidth

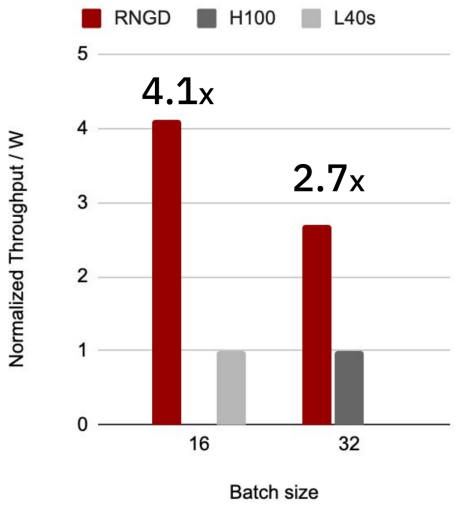
2x HBM3
CoWoS-S

Features For Cloud
Multiple-Instance support
Virtualization
Secure boot & model encryption

First token latency

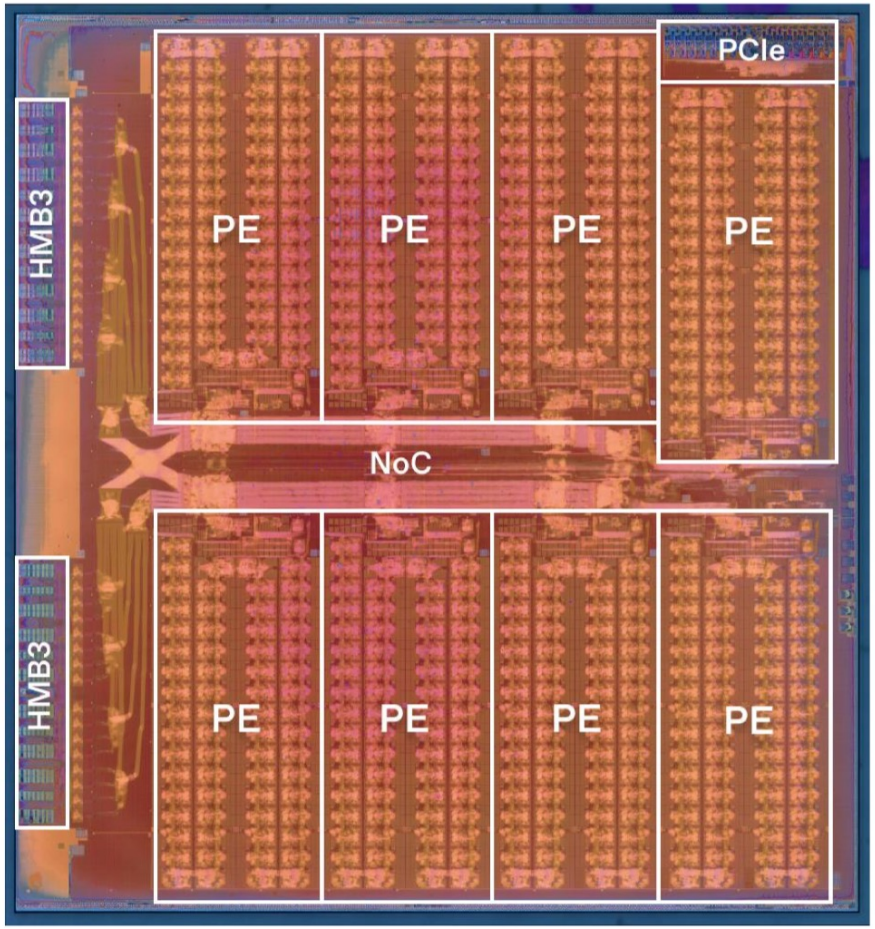


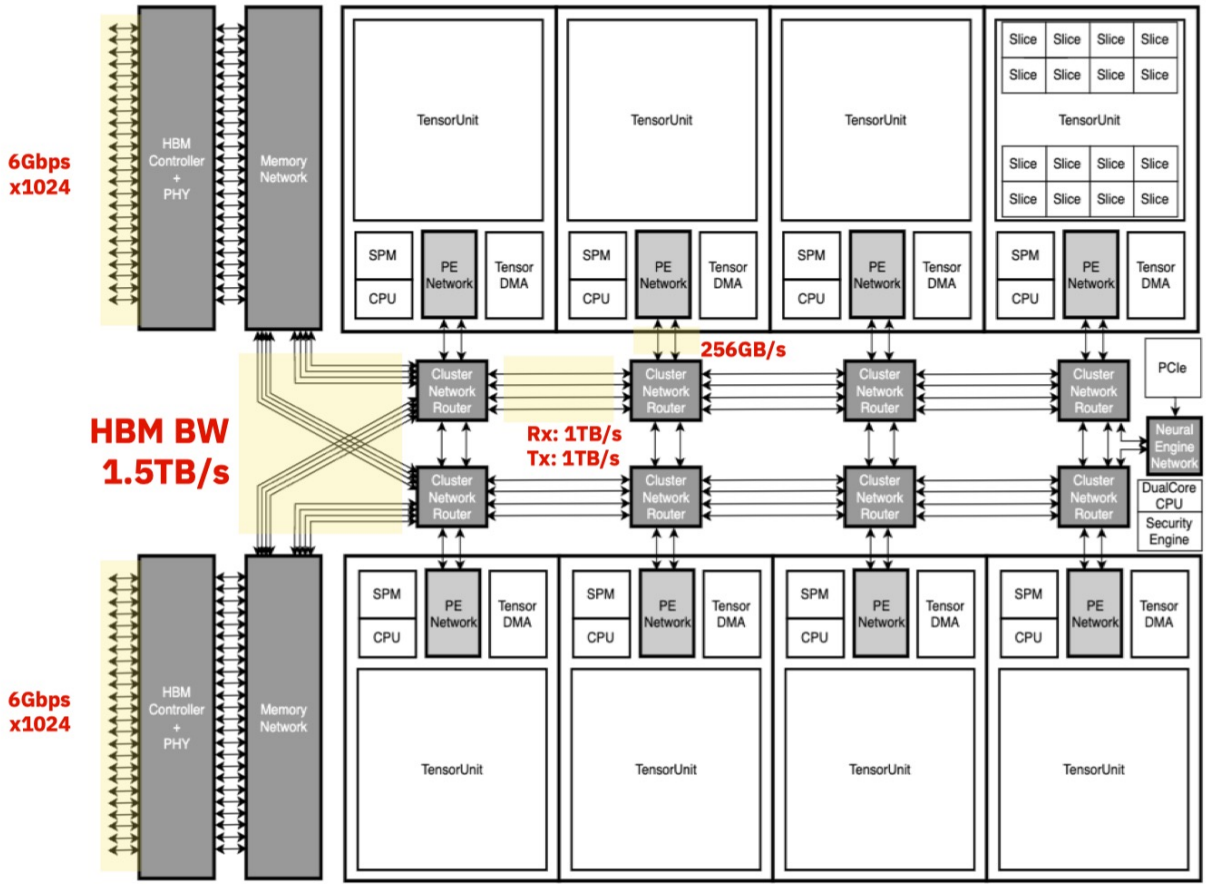
Throughput / W



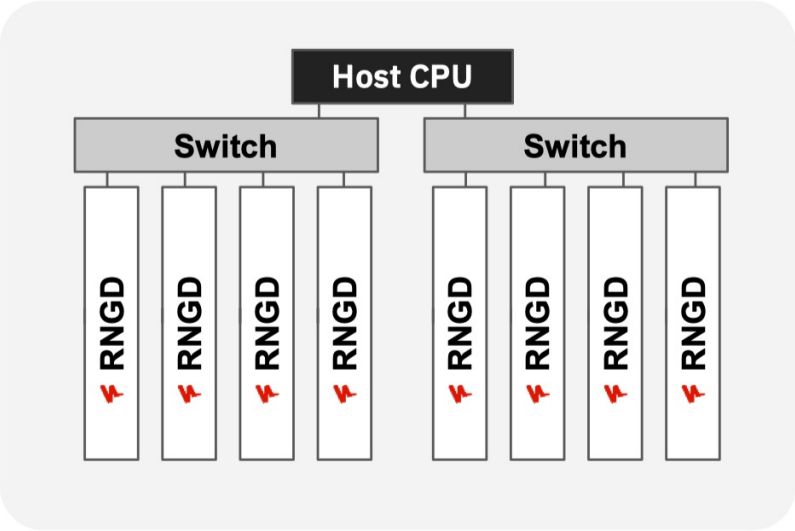
	RNGD	H100	L40s
Technology	TSMC 5nm	TSMC 4nm	TSMC 5nm
BF16/FP8 (TFLOPS)	256/512	989/1979	362/733
INT8/INT4 (TOPS)	512/1024	1979/-	733/733
Memory Capacity (GB)	48	80	48
Memory BW (TB/s)	1.5	3.35	0.86
Host I/F	Gen5 x 16	Gen5 x 16	Gen4 x 16
TDP (W)	150	700	350

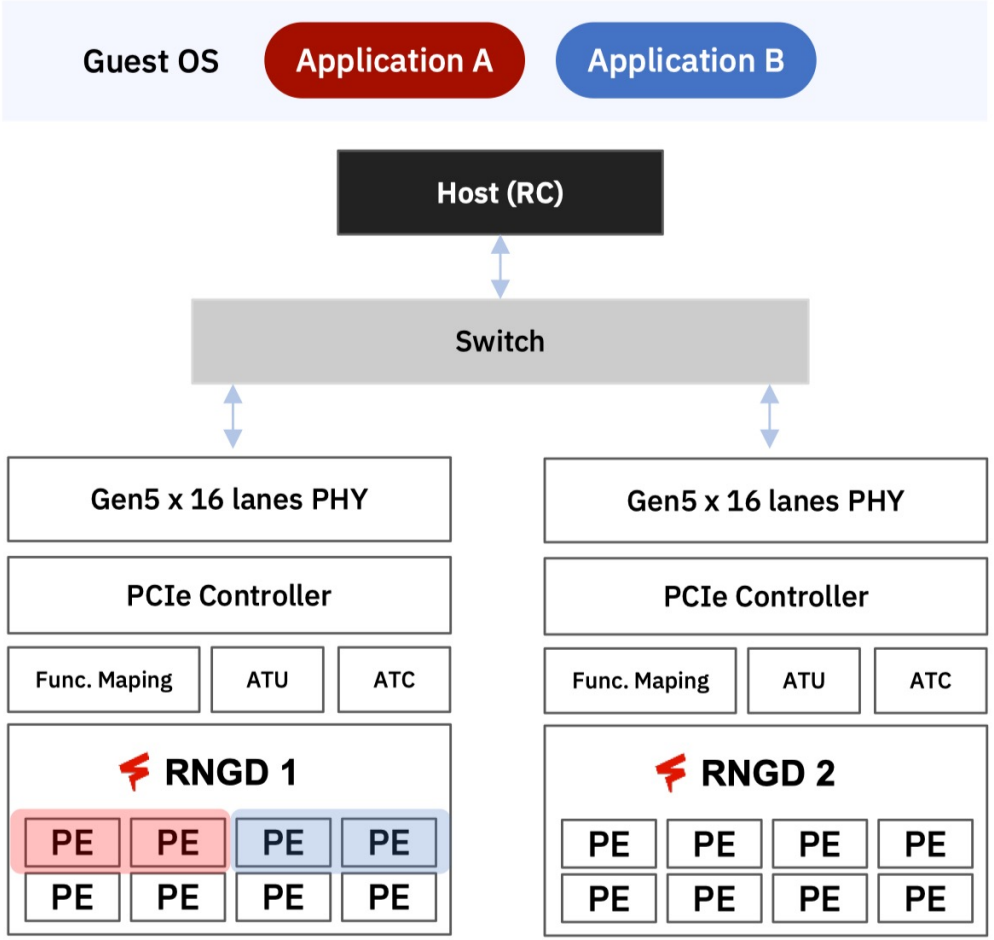
Interconnection Networks





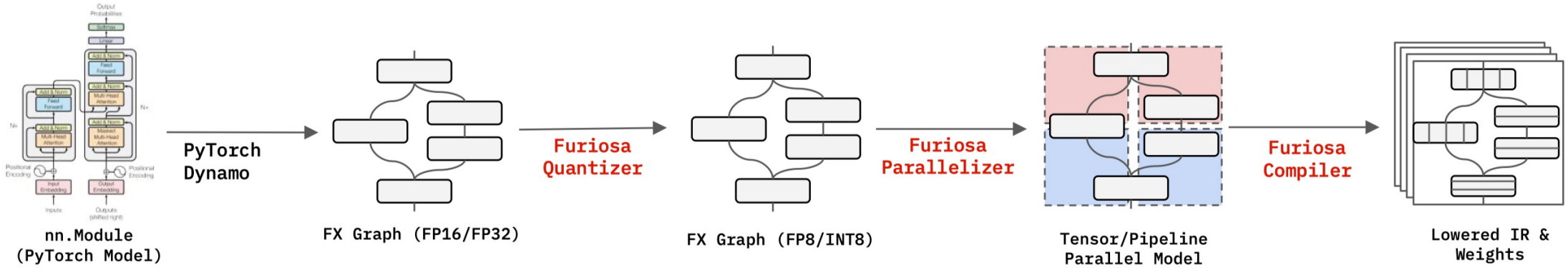
SR-IOV for Multi-Instance Support & Virtualization





- Supports up to 8 Virtual Functions for Virtual Machines
- 1, 2 or 4 PEs can be assigned for a VM

Furiosa LLM Stack



Furiosa LLM provides full-fledged features for LLM Serving, as following:

- PyTorch 2.0 integration
- Quantization toolkit with various quantization support (FP8, INT8, etc.)
- Tensor/Pipeline/Data parallelism over multiple cards
- Advanced DNN compiler
- State-of-the-art Serving optimization

Summary

raise the HW/SW interface by using tensor contraction as a primitive, a core computation in deep learning

- It enables Streamlined HW and maximize the parallelism and data reuse
- It offers flexibility to adequately support all deep learning models
- Compiler optimizes to minimize data movement across the entire model

Discussion

- Focus on GEMM, like a TPU with dataflow programming model and architecture
- Efficient data reuse in GEMM
- Reports on evaluation is insufficient
- Advanced process (5nm), bandwidth and computing power basically match H100 (1/2 bandwidth and ¼ computer power)

Meta MTIA

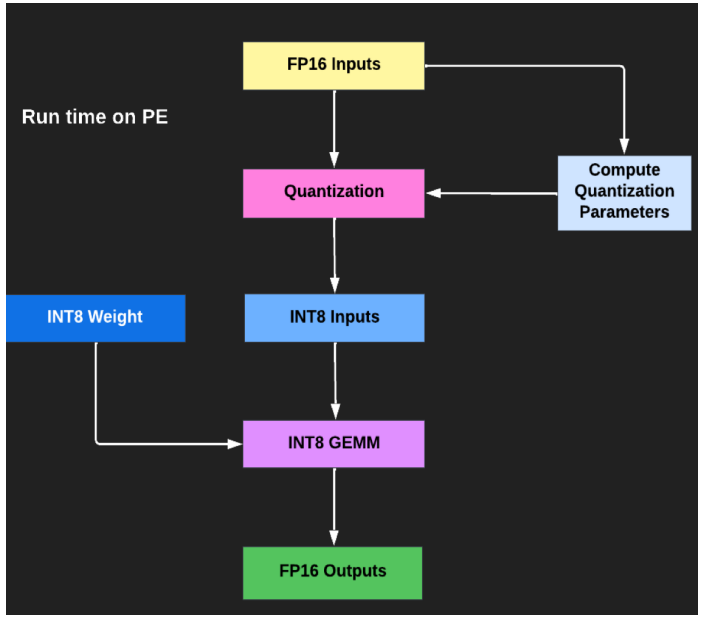
Two RISC-V cores

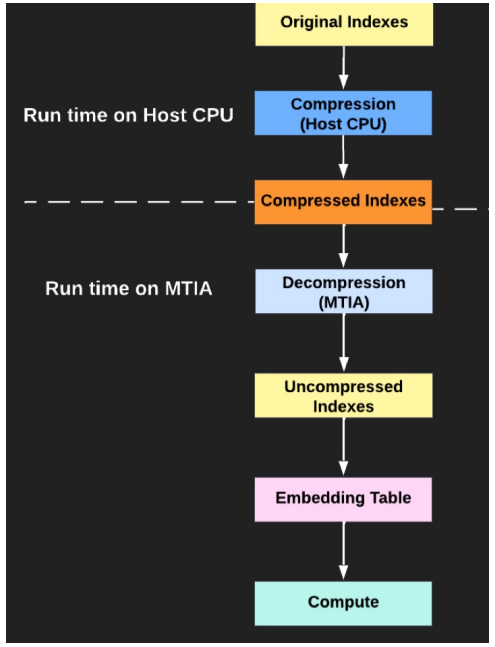
Integer Dynamic Quantization engine

- Collect min/max per batch during run time
- Support rowwise quantization
- Enable channel-wise symmetric dynamic quantization for FC operators
- Achieved over 99.95% accuracy comparing to baseline FP32 result

hardware decompression engine

- Added Decompression Engine to alleviate PCIe and network congestion
- Support for RFC1952 (GUNZIP/GZIP) standard encapsulating RFC1951 (Deflate Compression Format)
- Support for static and dynamic Huffman coded blocks





Others

Cooling tech

- Applications for thermo-electric cooling, Jesse Edwards, Phononic
- Next-Generation Cooling For NVIDIA Accelerated Computing, NVIDIA
- Thermal Techniques for Data Center Compute Density, Supermicro
- On-device AI and its thermal implications, Qualcomm

Dafalow

- Cerebras: break up HBM wall, WSE-3, 44GB of SRAM
- Sambanova SN40L RDU: operation fusion, one kernel call for all decoders
- TCP, RNGD: tensor contraction primitive and architecture

RISC-V core

- Xiangshan, Tensorrent, MTIA (one scale and one vector extension)

Datatype

- Maia 100-microsoft (MXFP4,6,9), B200-NVIDIA (MXFP, FP4)

Others

Takeaway

- thermal is a huge issue
- NVIDIA B200 -> a large chips, AMD MI300 -> 12 chiplets, advanced package technology