Microsoft

Inside Maia 100

Sherry Xu, Partner Lead SOC Architect

Chandru Ramakrishnan, Partner SW Eng Manager

Maia 100 Introduction

Microsoft's 1st-gen custom AI Accelerator

- Targets large-scale AI workloads
- Designed specifically for Azure to run production OpenAI models

Vertical integration to optimize performance and reduce cost

- Software-hardware codesign to unlock new capabilities
- Custom server boards with tailor-made racks
- Improve power efficiency

First generation designed for wide deployability

- Software stack build up
- Liquid cooling enablement

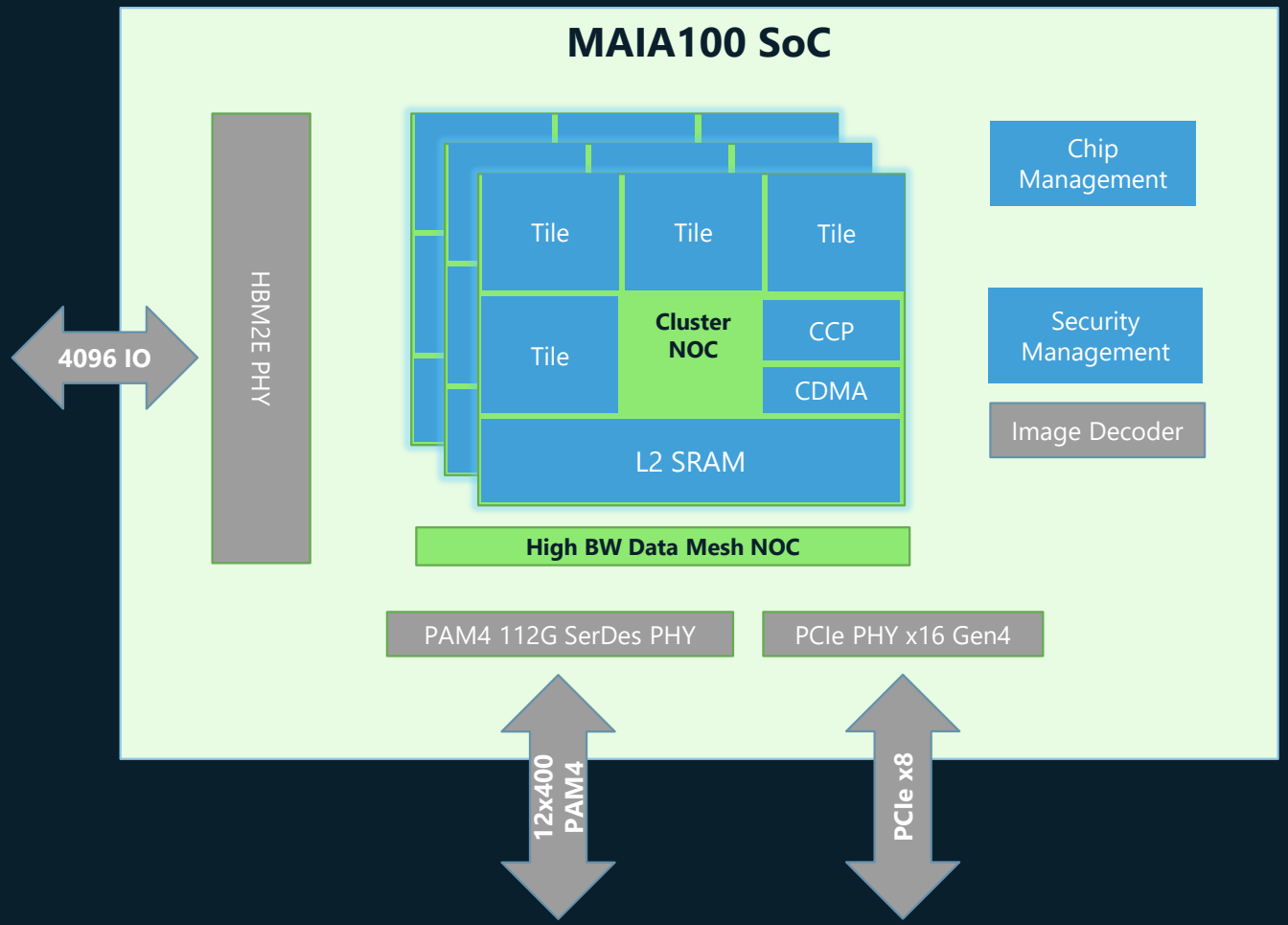
Maia 100 Specs

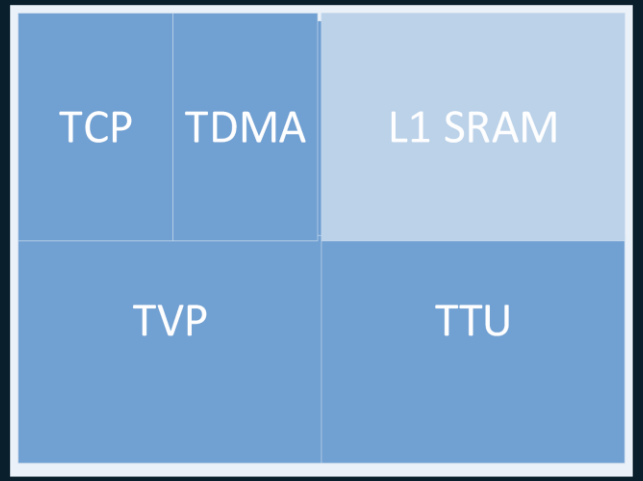
Maia 100 Specs

Chip Size	~820mm2@N5
Package/Interposer Technology	TSMC COWOS-S
HBM BW/Cap	1.8TB/s @64GB HBM2E
Peak Dense Tensor POPS	6bit: 3 9bit: 1.5 BF16: 0.8
L1/L2	~500MB
Backend Network BW	600GB/s (12x400gbe)
Host BW (PCIe)	32GB/s PCIe Gen5x8
Design to TDP	700W
Provision TDP	500W

Inside Maia 100

TTU: Tensor Unit
TVP: Vector engine
TDMA: Tile Data Movement Engine
TCP: Tile Control Processor





4 Tiles Per Cluster
16 Clusters Per Soc

ML-Specific Architecture

High speed tensor unit

- Supports a wide range of data types (including MX data format)
  - 9bit compute/6bit compute
- Constructed as an 16xRx16 unit
  - Sizing is trade-off between granularity loss and peak performance

Vector processor with custom ISA tailored for ML

- Loosely coupled superscalar engine
- Support FP32 & BF16

DMA engine supports different tensor sharding

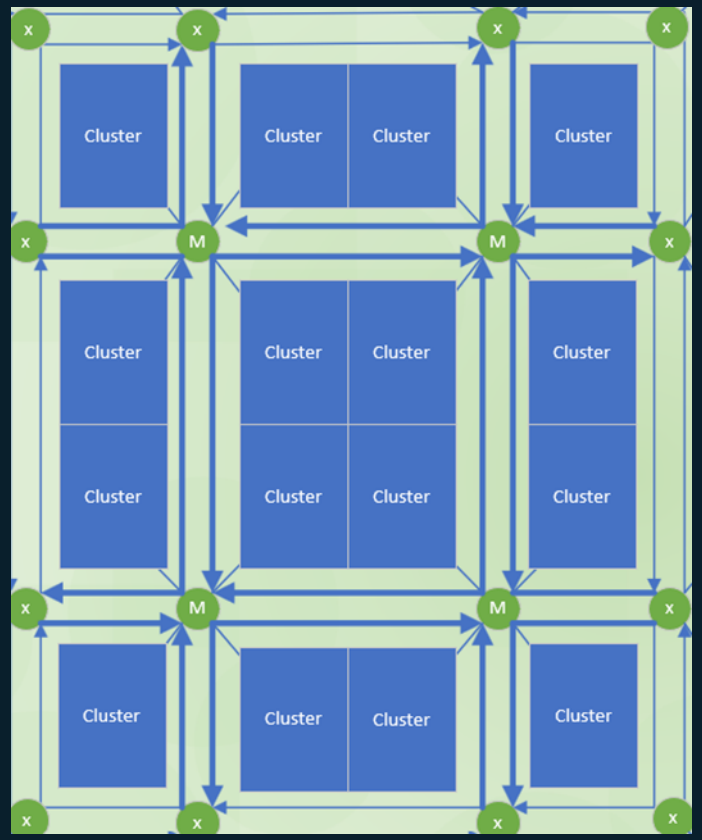
Hardware semaphore to enable async programming

Achieve great perf/W and performance

ML-Specific Architecture

Data movement is a well-known bottleneck for inference

- Data Compression and Narrow Data Type to improve storage and data movement
  - 4bit/6bit/9bit support
- Large L1/L2 scratch pad
  - Software-managed scratch pad
  - Improve data utilization & power efficiency
- Mesh-Like NOC topology optimizing for ML WL

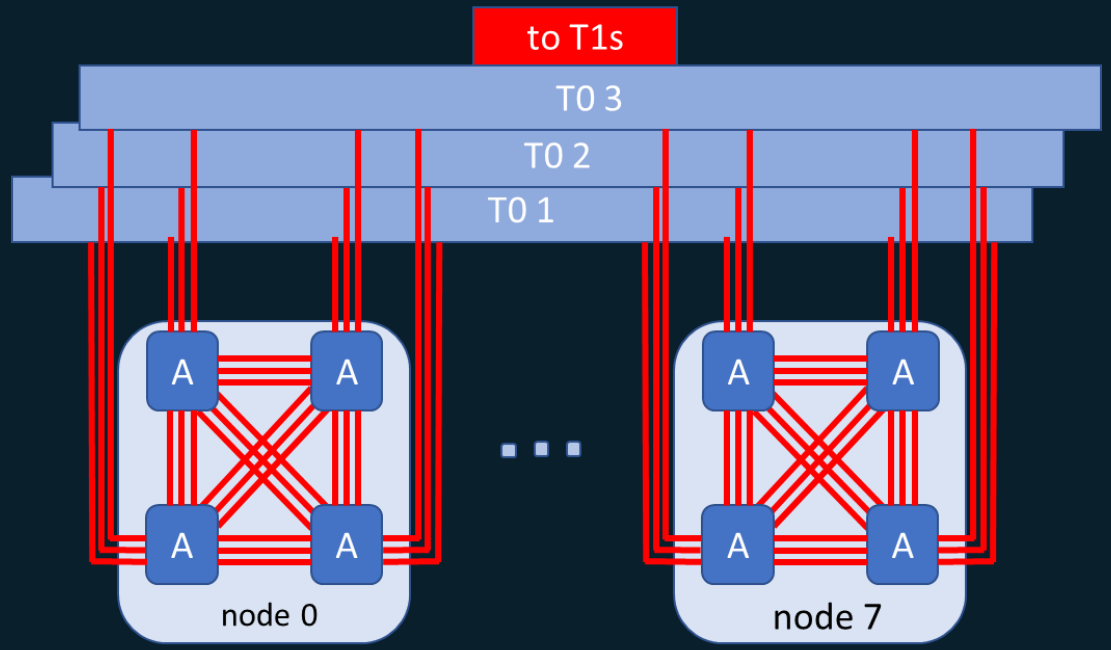


Achieve great perf/W and performance

Interconnect

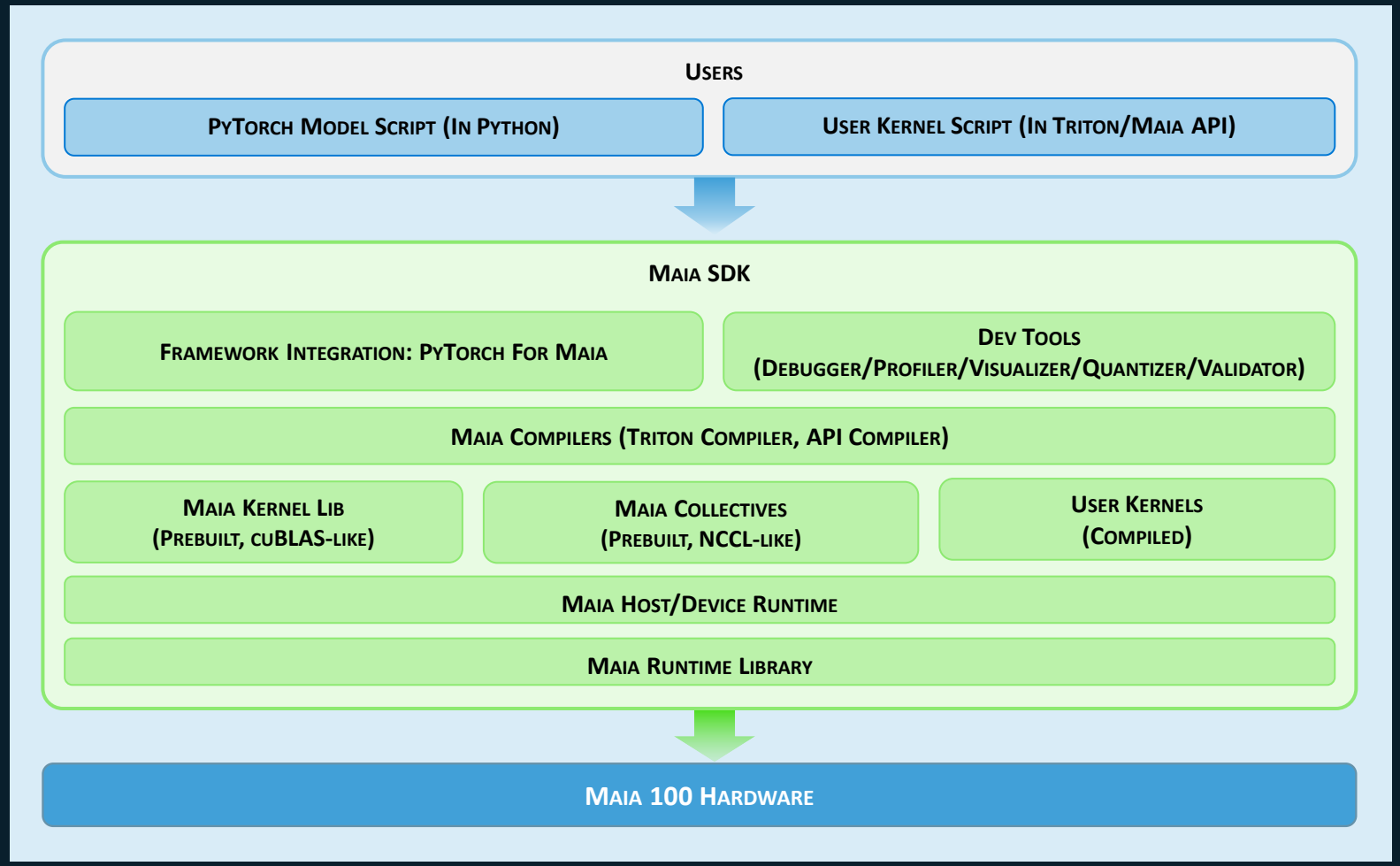
Supports Ethernet-based backend network with built-in encryption engine to help protect user data

- **High bandwidth Ethernet links**
  - All-gather/scatter-reduce at 4800Gbps
  - Any-to-any at 1200Gbps
- **Custom RoCE-like protocol**
  - Enhance Reliability & Load balance
  - AES-GCM encryption support
- **Unified Network**
  - Support direct and switch connectivity
  - Same network for both scale out and scale up



Maia SDK

Enables quick deployment of a model to Azure OpenAI Services



Framework integration

- Eager mode, graph mode, kernel dispatch

Developer tools

- Debugger, profiler, visualizer, model quantization and validation tools

Maia compilers

- Triton for agility and portability
- Maia API for highest performance

Maia kernel library (cuBLAS-like)

- Highly optimized implementation of the core set of ML compute kernels

Maia collectives (NCCL-like)

- Highly optimized implementation of communication kernels (e.g., AllReduce)

User kernels

- Compiled user kernels

Maia host/device runtime

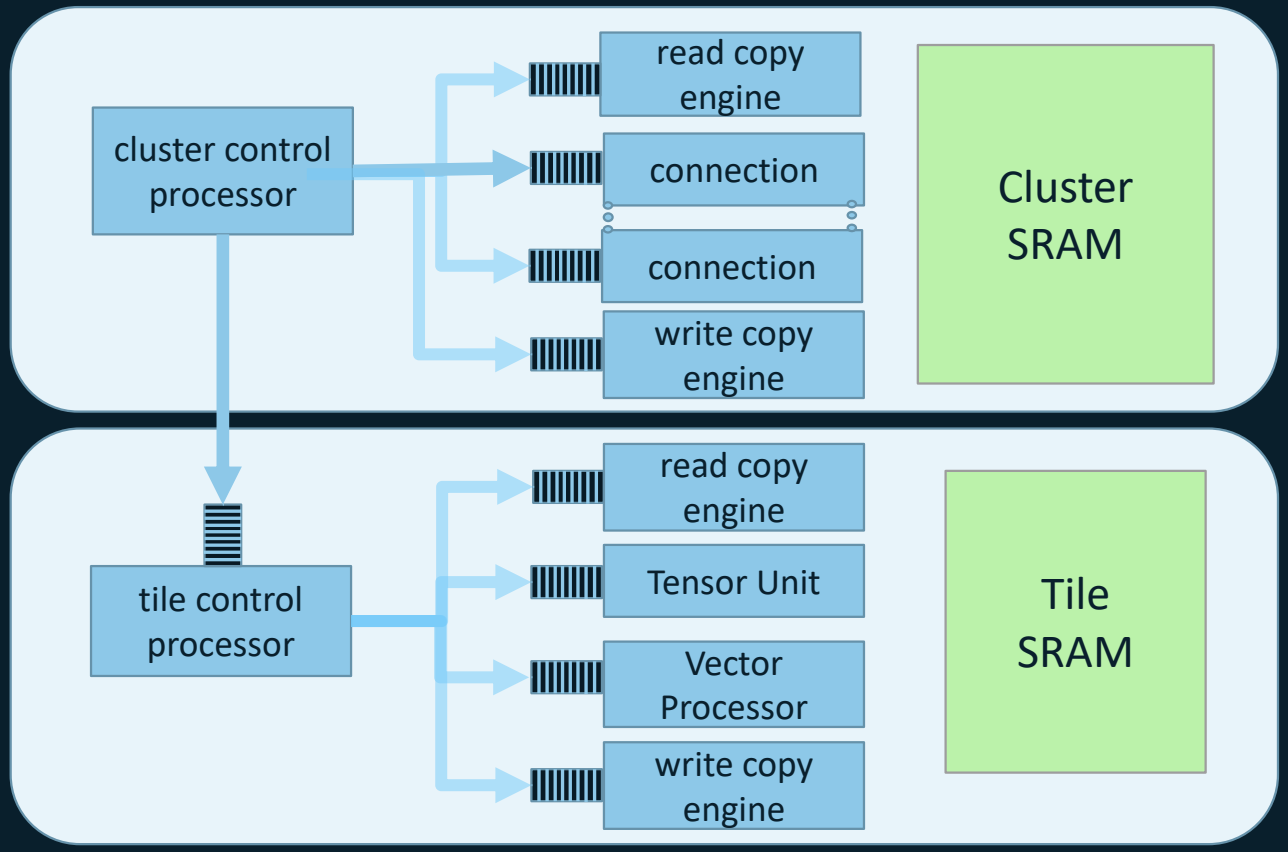
- Kernel invocation, device management

Maia Runtime Library (HAL)

- The set of hardware primitives exposed to software

Asynchronous Programming With Semaphores

Control processors send asynchronous commands to queues



Semaphores enforce command dependencies

```
Command ({in semaphores}, arguments,
{out semaphores})
```

1. command waits for in semaphores before execution
   - wait for input data to be ready and for output buffers available
2. command signals out semaphores after execution
   - signal that output data is ready and input buffers are available
3. control processor issues commands on control path
4. threads executing commands synchronize directly on the data path

Two Maia Kernel Programming Models

Triton	Maia API
High level	Low level
Flash-Attention in hundreds of lines of code	Flash-Attention in thousands of lines of code
Hardware agnostic	Maia-specific
Program Maia clusters, everything else is automatic	Individual programs for Maia clusters, tiles, vector processors
Compiler takes care of semaphores, memory, double buffering	Everything explicitly written in code

GEMM Partitioning & Scheduling

Gather based matrix multiplication

- Fuse element-wise activation function
- Overlap computation with network communication
- Send quantized data (MX) over network

Chips - 4 K = Input x WK									
M	2048			A Chips		A0	A1	A2	A3
K	12288								
N	12288		WK			N/4	N/4	N/4	N/4
					K/4	W00	W01	W02	W03
					K/4	W10	W11	W12	W13
					K/4	W20	W21	W22	W21
Input					K/4	W30	W31	W32	W33
	K/4	K/4	K/4	K/4					
M	X00	X01	X02	X03		Y0	Y1	Y2	Y3

		Maia 0				
						Weight
						W00
						W11
						W21
						W31
Phase 0	TTU	X00				Y00
						Y0 = Y00
	ANC(Data Transfer)	X00	X01			
						W10
Phase 1	TTU	X00	X01			Y01
	TVP				Y0 += Y01	
	ANC(Data Transfer)	X00	X01	X02		
						W20
Phase 2		X00	X01	X02		Y02
	TVP				Y0 += Y02	
	ANC(Data Transfer)	X00	X01	X02	X03	
						W30
Phase 3		X00	X01	X02	X03	Y03
	TVP				Y0 += Y03	
Output		Y0 = Y00 + Y01 + Y02 + Y03				

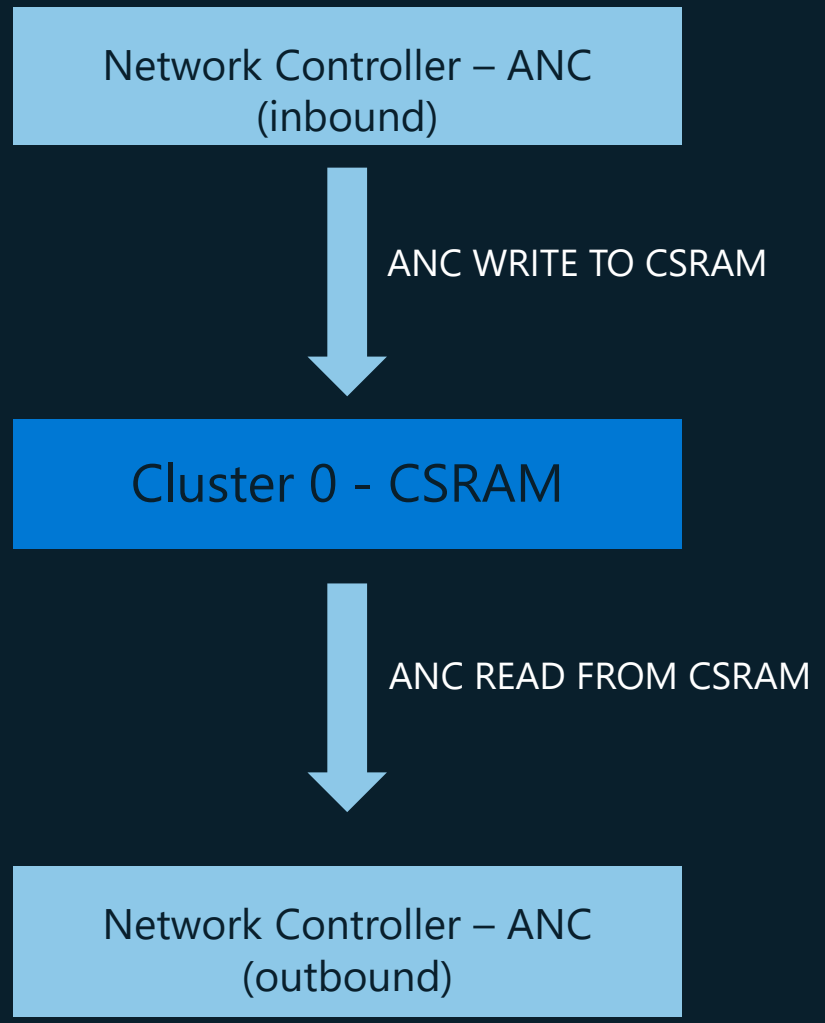
GEMM Partitioning & Scheduling

Optimized on-chip data flow

- Computation inside cluster
- Load weights from local HBM
- On-chip activation gather through NOC Inner-Ring
- Network activation gather through 9 intra-node links

Leverage L2 SRAM (CSRAM)

- To buffer activations and intermediate results (high precision)
- To reduce HBM accesses for network traffic



PyTorch Integration

Out-of-box experience with PyTorch models

- Execute PyTorch model with single line change
- Maia PyTorch backend
  - Resource management (HBM / SRAM / Stream / Event)
  - Eager mode execution
    - Support set of PyTorch operators that are used in popular LLMs / SLMs with a high performance kernel library
  - Graph mode execution
    - Integrate with torch.Compile ecosystem

Import maia backend

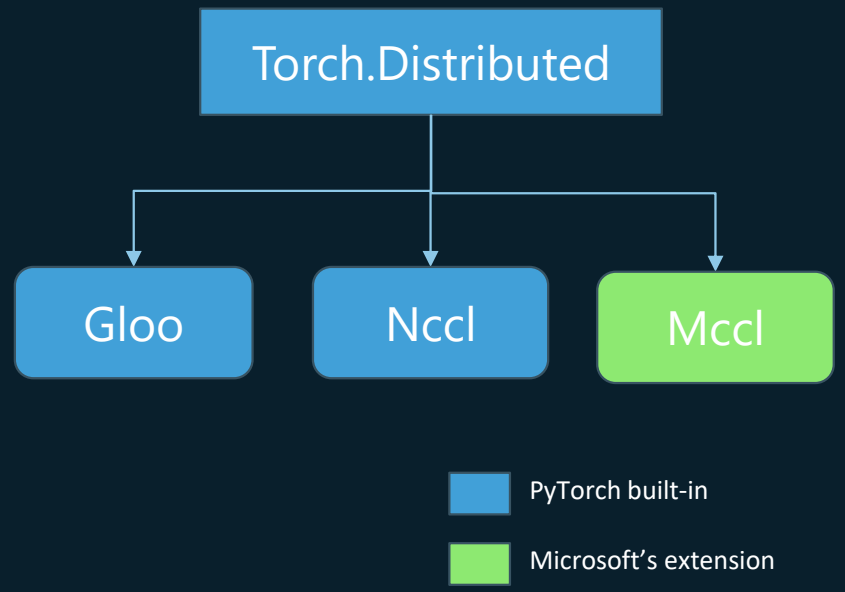
```
import torch
import torch.maia

from transformers import AutoModelForCausalLM, AutoTokenizer
~~device = "cuda"~~ # the device to load the model onto
device = "maia"


model = AutoModelForCausalLM.from_pretrained("mistralai/Mistral-7B-v0.1")
tokenizer = AutoTokenizer.from_pretrained("mistralai/Mistral-7B-v0.1")

prompt = "My favorite condiment is"

model_inputs = tokenizer([prompt], return_tensors="pt").to(device)
model.to(device)
generated_ids = model.generate(**model_inputs, max_new_tokens=100, do_sample=True)
print(tokenizer.batch_decode(generated_ids)[0])
```

Switch to Maia device

Communication Library

- Used for models that require > 1 device
- MCCL: Maia's TorchDistributed backend

Operators	Gloo	Nccl	Mccl
send	✓	✓	✓
recv	✓	✓	✓
broad_cast	✓	✓	✓
all_reduce	✓	✓	✓
reduce	✓	✓	✓
all_gather	✓	✓	✓
gather	✓	✓	✓
scatter	✓	✓	✓
reduce_scatter	✗	✓	✓
all_to_all	✗	✓	✓
barrier	✓	✓	✓



Developer Tools in the Maia SDK

Maia Device Management

- ***maia-smi***

Development and Debugging

- **Maia Debugger**
  - maia-gdb cmdline debugging experience
  - VS Code plugin – integrated enhanced debugging experience
- **Crash Dump Analysis**
- **Virtual Platform (maia-sim)** –Maia device emulator to enable kernel development on Maia locally
- Race Condition Analysis, Execution Graph Analysis
- Logging, Profiling Library

Performance and Monitoring

- Maia Profiling Tool (***maia-prof***)
- Maia Resource Usage Monitoring Tool (***maia-mon***)

Discussion