FURIOSA

Beyond *Just* Hardware

Full-stack Optimization Towards Efficient AI Inference

Hyunsik Choi, *Head of SW Platform,* Jihoon Yoon, *Product Marketing Manager*

2017-2021

FuriosaAI founded &
Launch Gen 1 vision NPU

2021

GPT3 inspired
RNGD

2022

RNGD Development
Kick off

2024 May

RNGD raw silicon
sample arrival

2024 July

First LLM demo

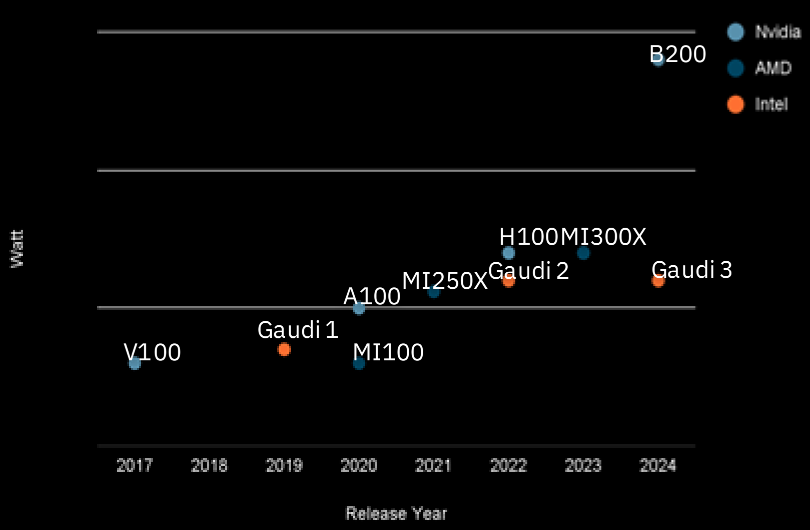
Key Points

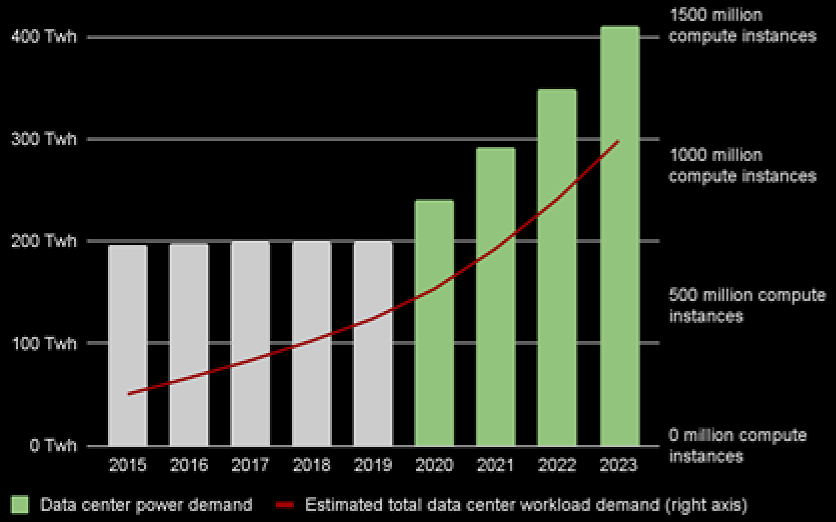
01 Mass AI adoption is bottlenecked

02 Energy efficient AI inference

03 Full-stack optimization for achieving efficiency

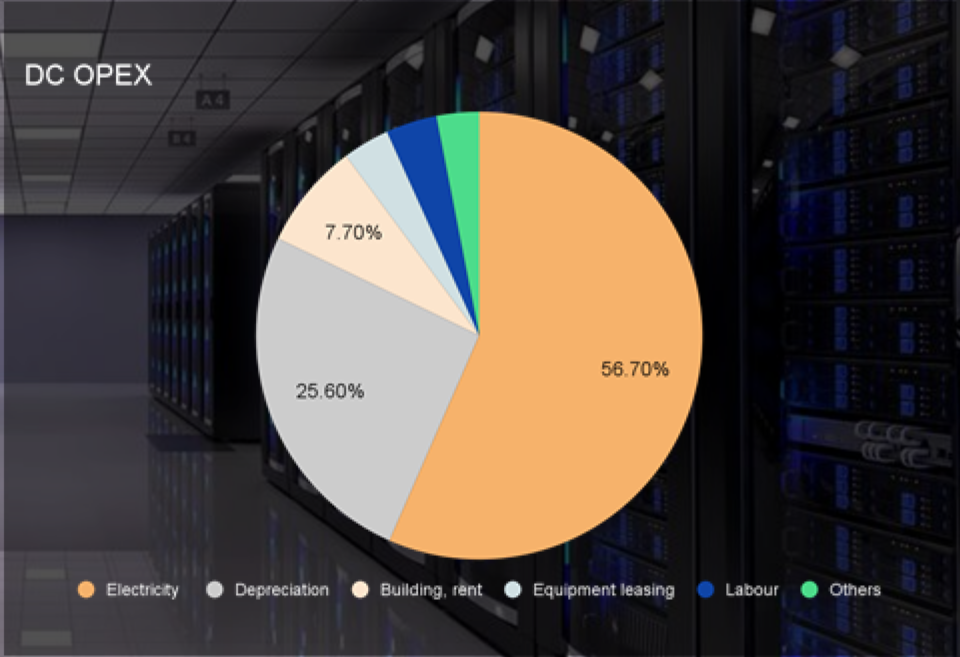
AI has broken energy efficiency





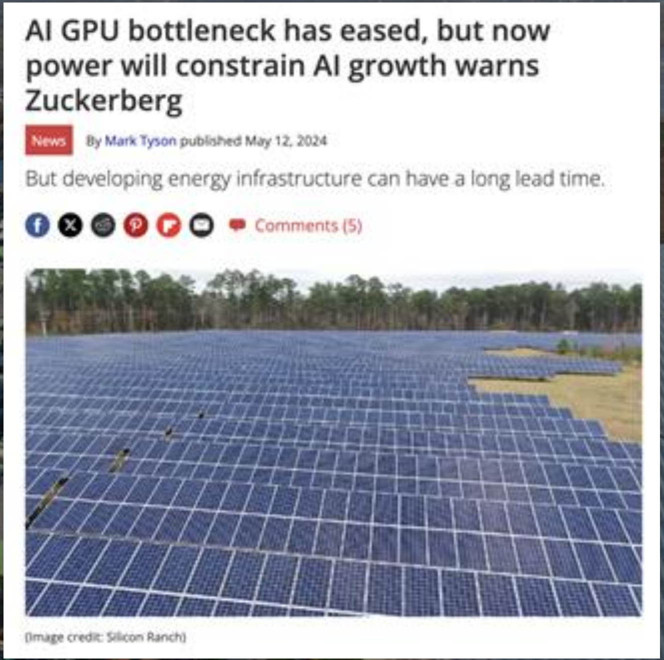
Source: Masanet et al. (2020), Cisco, IEA, Goldman Sachs Research

Electricity is already a huge *financial* and env*ironmental* burden on data centers



Source: HARTING White Paper (2024)

AI inference will be *everywhere*.
But is our infrastructure ready?



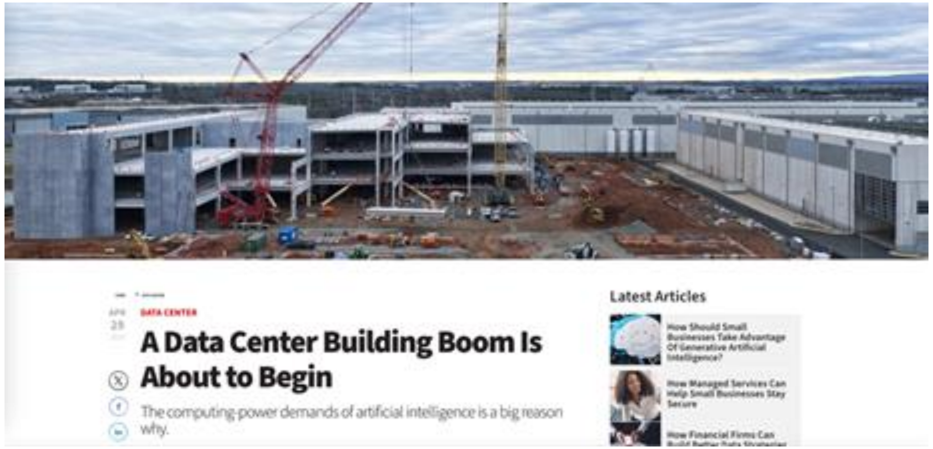
AI GPU bottleneck has eased, but now power will constrain AI growth warns Zuckerberg

News By Mark Tyson published May 12, 2024

But developing energy infrastructure can have a long lead time.

Comments (5)

(Image credit: Silicon Ranch)



APR 29

DATA CENTER

A Data Center Building Boom Is About to Begin

The computing-power demands of artificial intelligence is a big reason why.

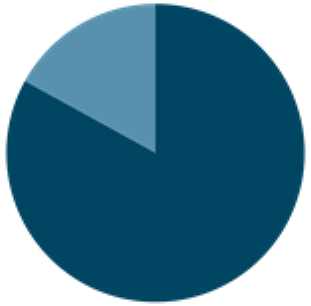
Latest Articles

How Should Small Businesses Take Advantage Of Generative Artificial Intelligence?

How Managed Services Can Help Small Businesses Stay Secure

How Financial Firms Can Build Better Data Strategies

Data center cooling infrastructure (2024)

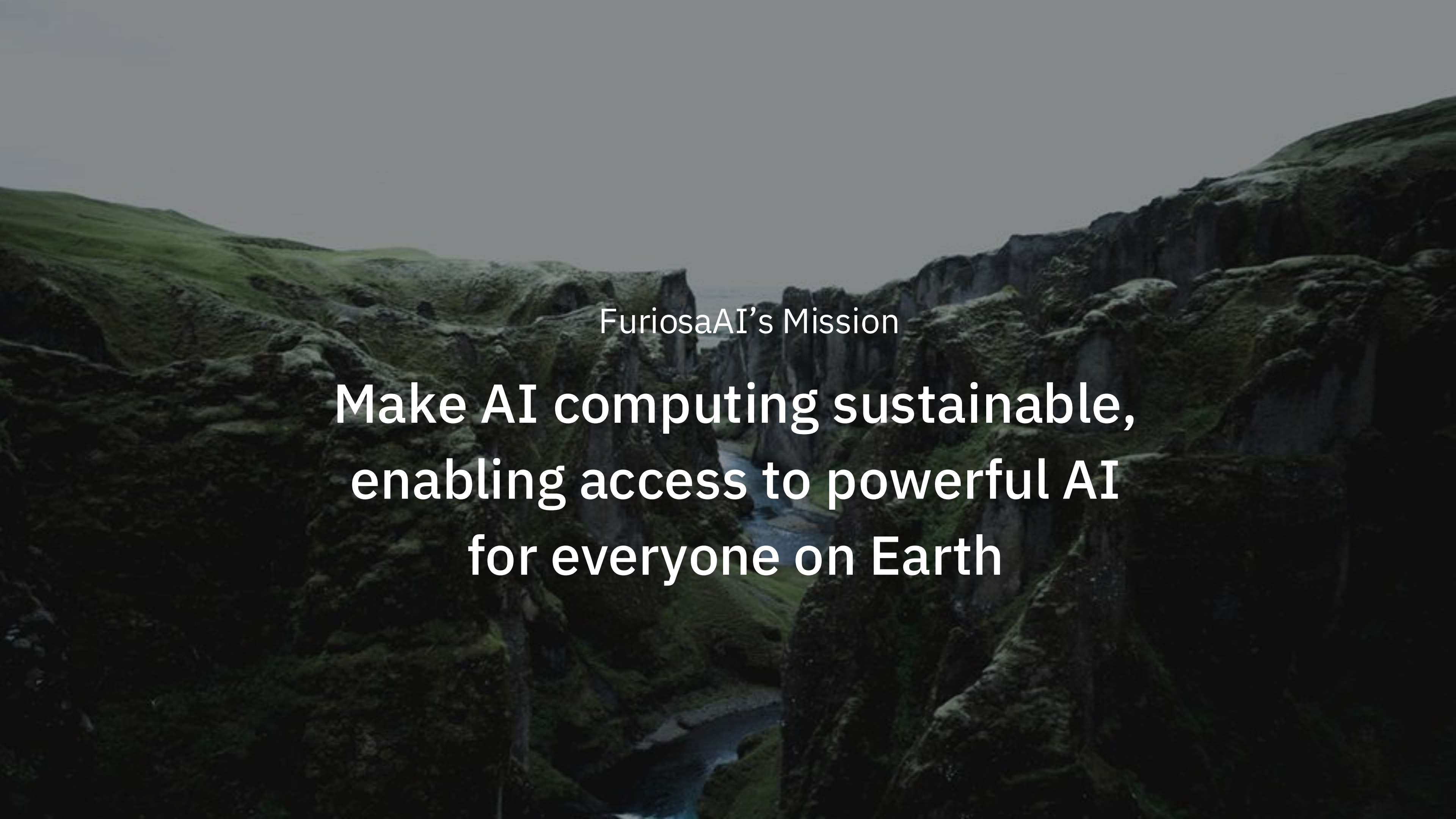


"Average server rack densities are increasing but **remain below 8 kW.** The majority of facilities do **not have racks above 30 kW**, and those that do have only a few."

- Uptime Institute Global Datacenter Summary 2024

What if

there is a more energy *efficient AI inference* solutions that can be deployed *anywhere within existing infrastructure*.



FuriosaAI’s Mission

Make AI computing sustainable, enabling access to powerful AI for everyone on Earth

RNGD: Powerfully Efficient AI Inference

Data center AI accelerator built for the era of LLM
and other generative AI models

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 data centers

PCIe P2P support For LLMs

256 MB SRAM
384 TB/s On-chip Bandwidth

2 x HBM3
CoWoS-S

Features For Cloud

Multiple-Instance support
Virtualization
Secure boot & model encryption

Early performance numbers:
60% higher perf/watt than current inference solutions

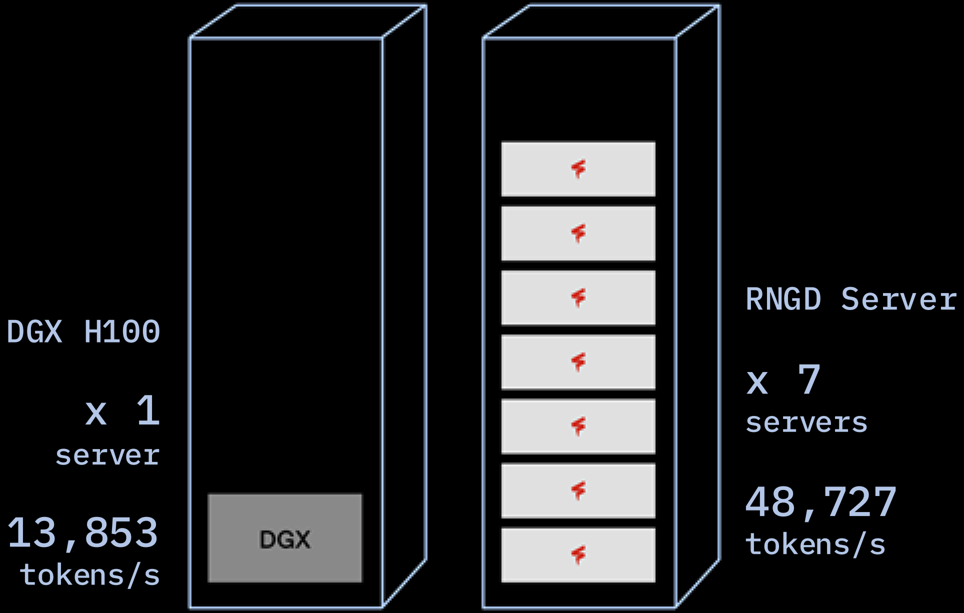
GPT-J 6B MLPerf Benchmark Scenario (99% accuracy)

	RNGD	NVIDIA L40S	Intel Gaudi 2	Google TPU v5e
Performance (queries / sec)	11.5 (FP8)	12.3 (FP8)	10.51	2.5
Power (watt)	185	320	Unknown	Unknown
Data source	measured	measured	MLPerf 3.1	MLPerf 4.0

Disclaimer: As of Aug 2024, unverified by MLPerf

3.5x *compute per rack*

Lower total cost of ownership, with less energy usage and fewer racks. Compatible with air-cooled data centers of today

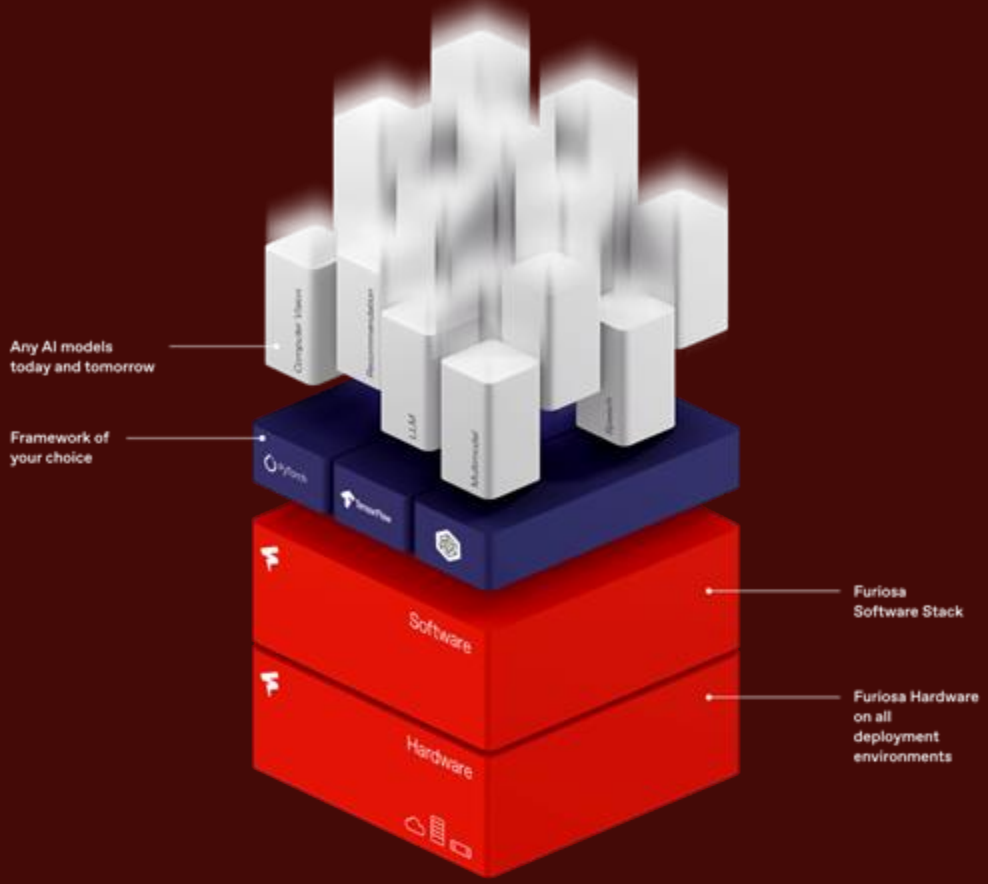


Most data center racks today are below 15kW

data above is for running Llama 3 70 B

Beyond hardware

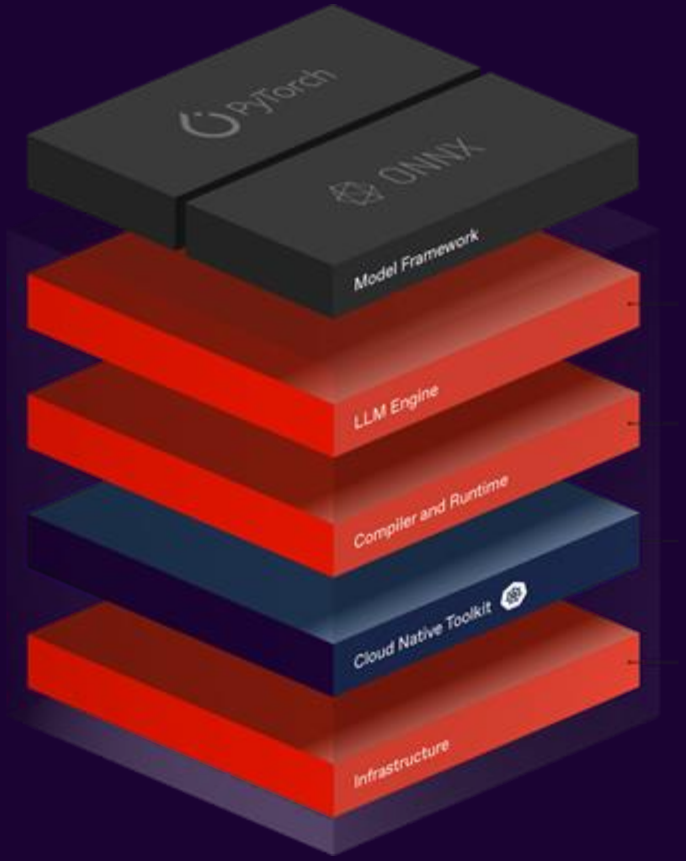
Full-stack innovation and optimization for maximized efficiency in AI inference



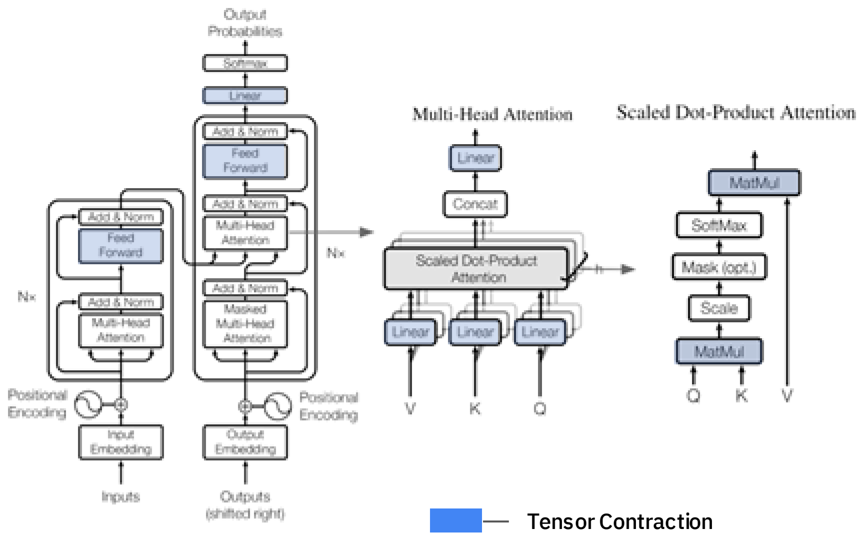
Model Execution

Serving

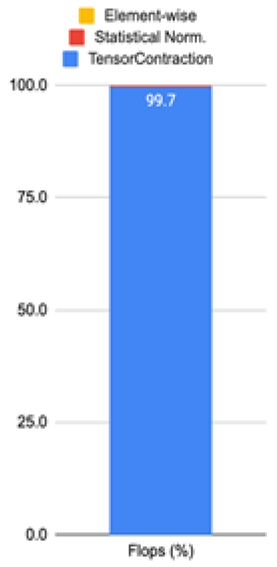
Utilization



Tensor Contraction, The Core Computation in Deep Learning



🟦 — Tensor Contraction



Flop analysis for BERT*

"Data movement is the major bottleneck for efficiency"

Source: "Data Movement is All You Need," MLSYS'21

TCP: A Tensor Contraction Processor for AI Workloads

ISCA 2024 Submission #23 – Confidential Draft – Do NOT Distribute!!

Abstract—We introduce a novel tensor contraction processor (TCP) architecture that offers a paradigm shift from traditional architectures that rely on fixed-size matrix multiplications. TCP aims at exploiting the rich parallelism and data locality inherent in tensor contractions, thereby enhancing both efficiency and performance of AI workloads.

TCP is composed of coarse-grained processing elements (PEs) to simplify software development. In order to efficiently process operations with diverse tensor shapes, the PEs are designed to be flexible enough to be utilized as a large-scale single unit or a set of independently running small compute units. We aim at maximizing data reuse on both levels of inter and intra compute units. To do that, we propose a circuit switch-based fetch network to flexibly connect compute units to enable inter-compute unit data reuse. We also exploit input broadcast to multiple contraction engines and input buffer based reuse to further exploit reuse behavior in tensor contraction. Our compiler explores the design space of tensor contraction considering tensor shapes and the order of their associated loop operations as well as the underlying accelerator architecture.

A TCP chip was designed and fabricated in 5nm technology, offering 256/512/1024 TOPS (BF16/FP8 or INT8/INT4) with 256 MB SRAM and 1.5 TB/s 48 GB HBM3 under 150 W TDP. We

there is inherently high complexity in parallelizing operations while being NoC-aware. Moreover, generating programs for numerous wimpy cores is known to be more challenging than for a few brawny cores [6]. Except for GPUs, there are few successful commercial chips with a large number of small PEs that provide a software stack capable of compiling arbitrary tensor-manipulating programs into efficient executables.

Inference is often characterized by diverse tensor shapes and thus it is essential to exploit the parallelism and data reuse derived from the tensor shapes as well as the batch size. Thus, in case of large matrix unit based chips, it is challenging to fully utilize the large matrix units across various shapes and types of tensor operations [5], [22], [24].

Instead of matrix multiplications, we use tensor contraction as a primitive. This approach not only enables massively parallel operations but also incorporates pipelining over the time axis, similar to vector processors. We have designed large coarse-grained processing elements (PEs) which can be split into smaller compute units called *slices*, as illustrated in

…wing for more flexible configurations for diverse

…s. Depending on the setup of the fetch network

…the slices, the entire set of slices can function as

…ocessing element or individual slices can operate

…dependent, and parallel compute units.

…e, in the case of the attention layer of transformer

…, a PE's slices can be configured to operate in

…each head, e.g., 16 slices per head (see Fig. 10).

…ntinuously fetched in a pipelined manner through

…work, allow the operation units to be utilized at

…put. This enables us to adopt various data reuse

…t efficiently utilize the limited input/weight/output

…slices as demonstrated in the case study of the

…B model execution (Section VI).

…operation units perform computations deterministic…

…fined by the software, TCP achieves predictable

…. This enables us to develop accurate cost models

…nce and energy consumption. Our compiler lever…

…models when exploring possible configurations of

…or shapes and their contraction orders.

…mainder of the paper, we first explain our low-level

…ns of tensors and operations in Section II. Then,

… the chip-level architecture and micro-architecture

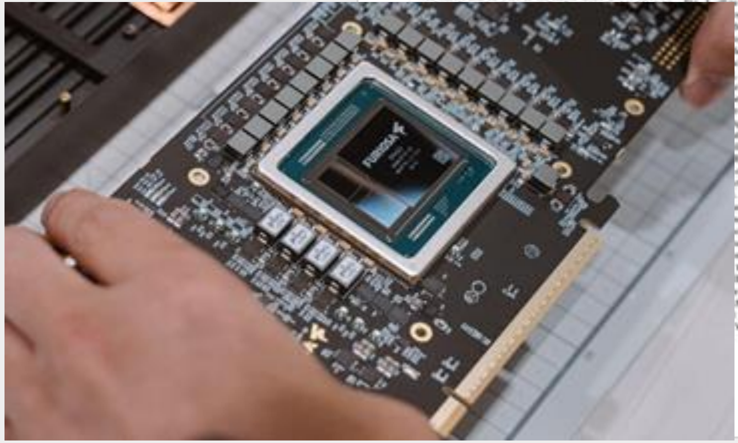
…III and IV. We explain our programming interface

… stack in Section V. Finally, we show how a

…hine learning model such as LLaMA-2 7B can

… on our chip, discuss the performance results

…VI, and share our lessons learned during the



TCP (Tensor Contraction Processor)

"TCP aims at **exploiting the rich parallelism and data locality** inherent in tensor contractions, thereby enhancing both efficiency and performance of AI workloads."

TCP: A Tensor Contraction Processor for AI Workloads
Presented at ISCA: International Symposium on Computer Architecture, 2024

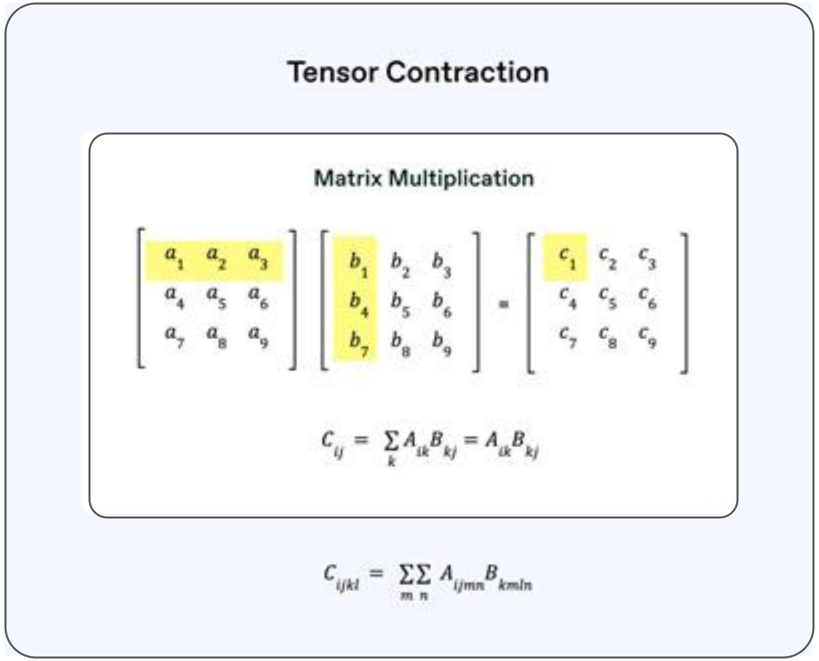
Tensor Contraction, *not* Matmul, as a Primitive

Tensor contraction is a higher dimensional generalization of matrix multiplication.

Tensor contraction is declarative

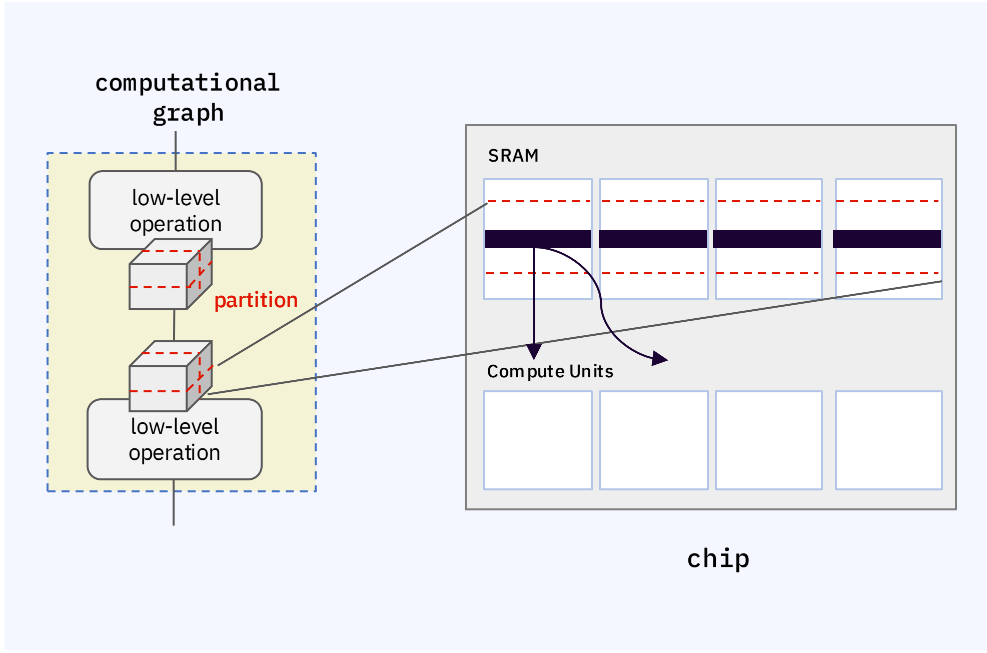
No explicit memory layout for data

No explicit scheduling for computation


Tensor Contraction
Matrix Multiplication
$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$
$C_{ij} = \sum_k A_{ik} B_{kj} = A_{ik} B_{kj}$
$C_{ijkl} = \sum_m \sum_n A_{ijmn} B_{kmln}$

DNN Graph Compiler: End-to-End Model Efficiency

- Optimal memory layout and operation scheduling for maximum data reusability
- Temporal pipeline opportunities
- Operator fusion and memory allocation, split/merge scheduling

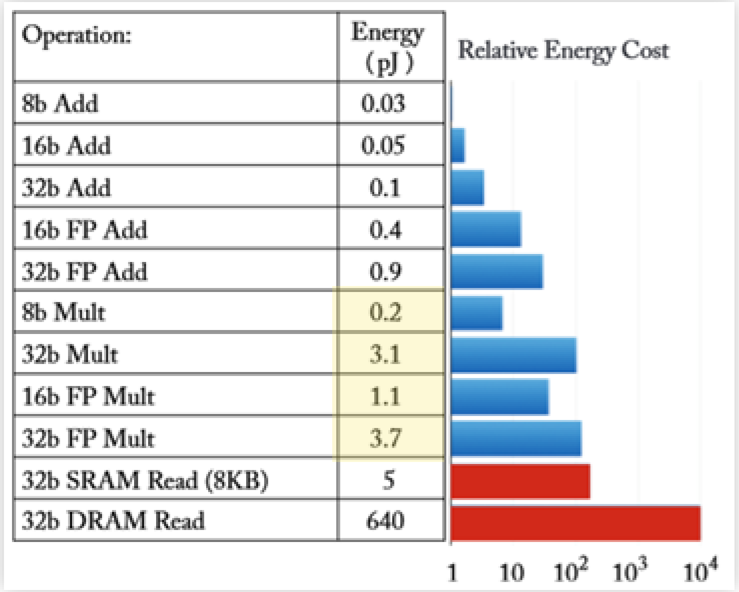


Quantization Becomes More Critical as Model Sizes Grow

Efficiency gains through quantization

- Inference latency
- Computation time
- Memory footprint
- Energy consumption

Energy Consumption
(Numbers are rough approximations for 45nm)



Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Mult	0.2
32b Mult	3.1
16b FP Mult	1.1
32b FP Mult	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640

Relative Energy Cost
1
10
10^2
10^3
10^4

Computing's Energy Problem, M. Horowitz, ISSCC, 2014
Slide: Courtesy of Prof. Shao

Furiosa Quantizer: Graph-Based Automated Tool

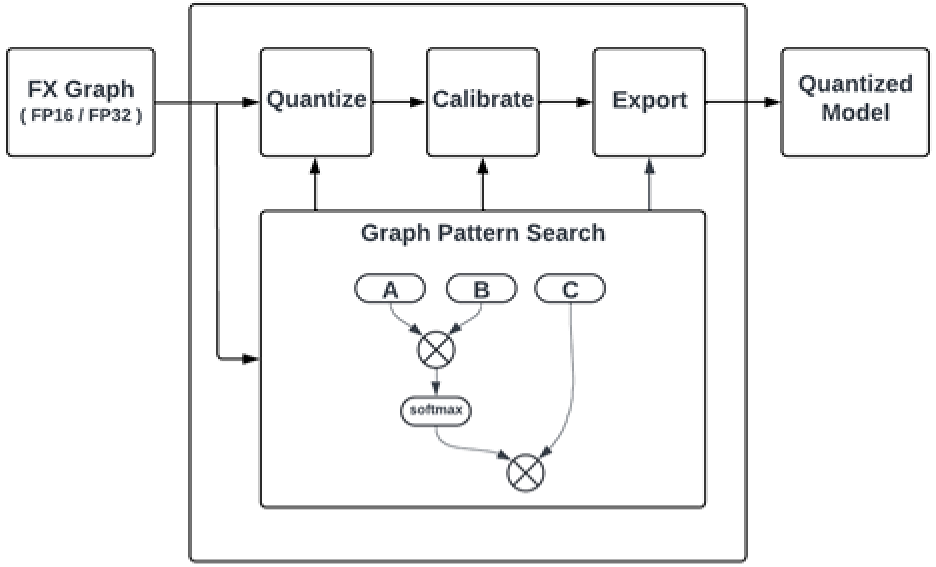
End-to-end automated quantization

Supports arbitrary customized LLM models using graph pattern search

BF16, INT8 Weight-Only (W8A16),

FP8 (W8A8), INT8 SmoothQuant (W8A8), INT4 Weight-Only (W4A16 AWQ / GPTQ)

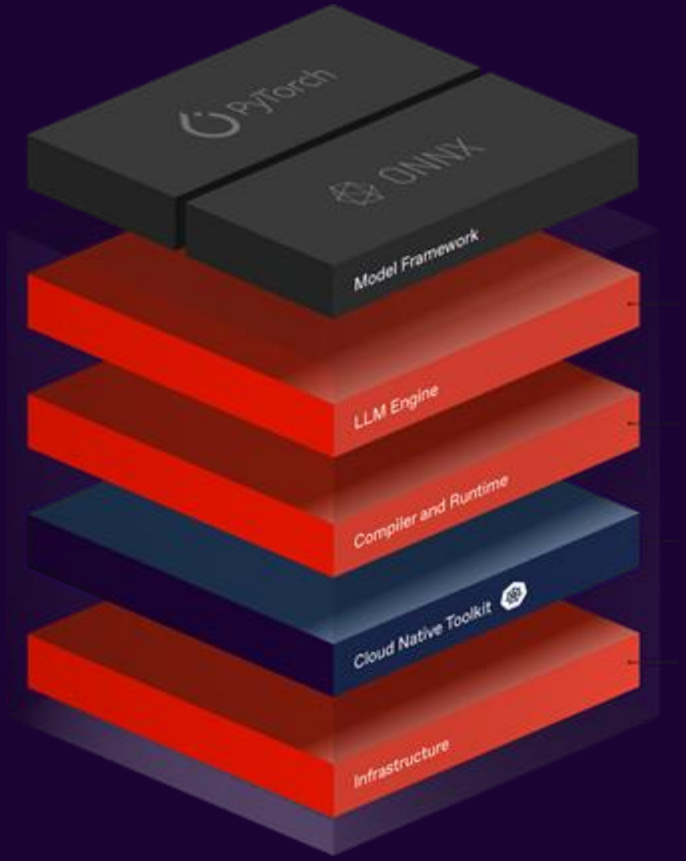
Furiosa Quantizer



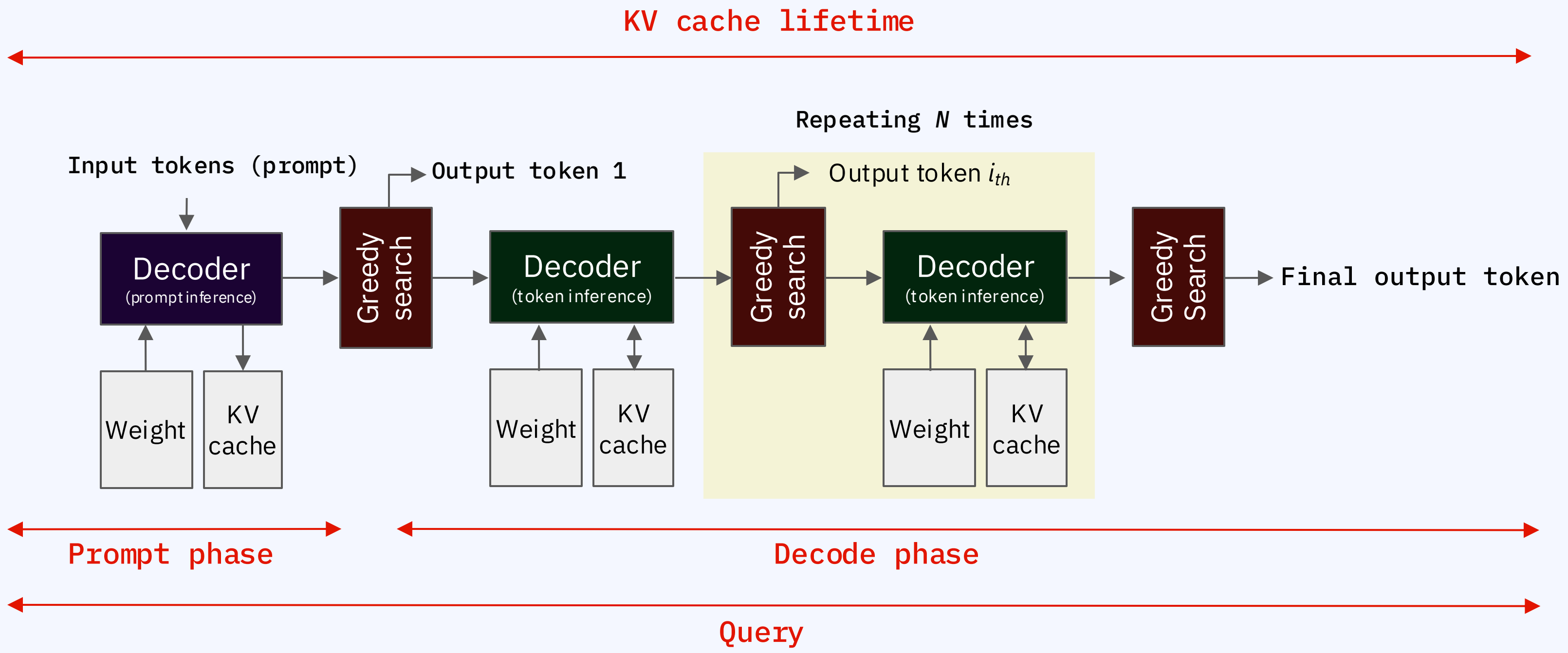
Model Execution

Serving

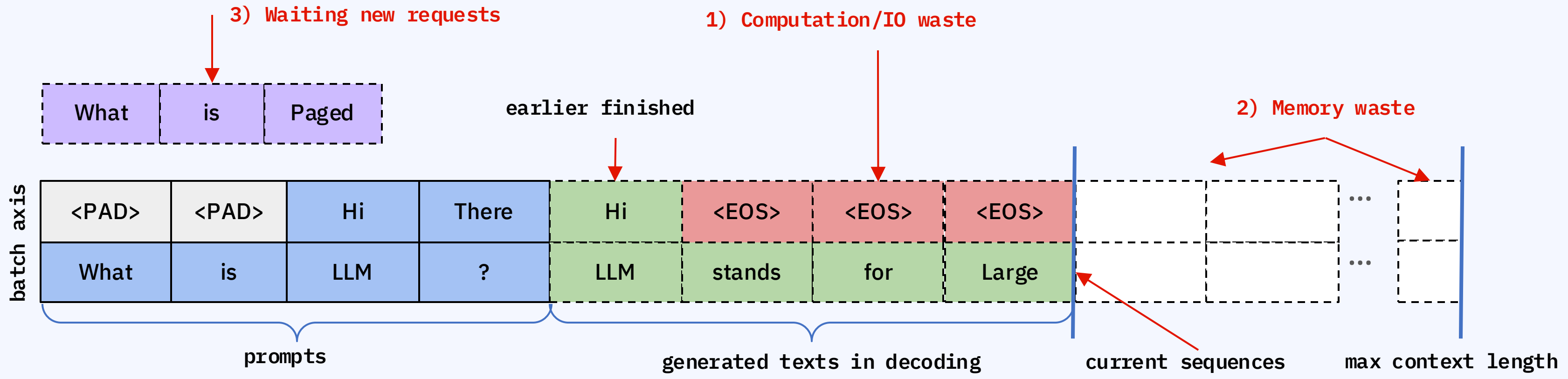
Utilization



Generative Inference Basics



Challenges in Generative Model Serving



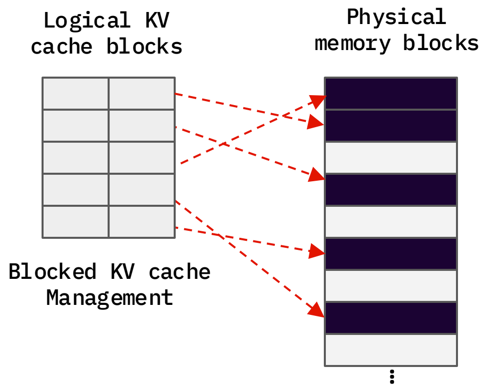
Challenges of auto-regressive execution in serving

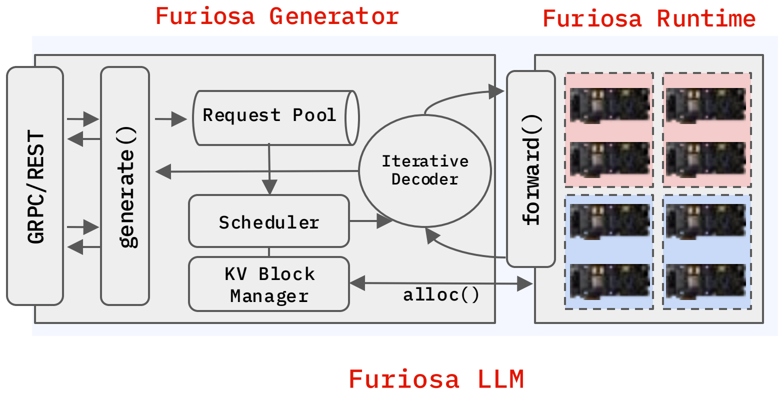
Furiosa LLM: High-throughput Serving Engine for LLMs

High throughput serving with SOTA optimization

- Continuous batching allows immediately starting incoming requests when resource is available.
- PagedAttention eliminates compute and IO waste
- Blocked KV cache reduces significantly memory wastes

6x *Increase in inference performance*

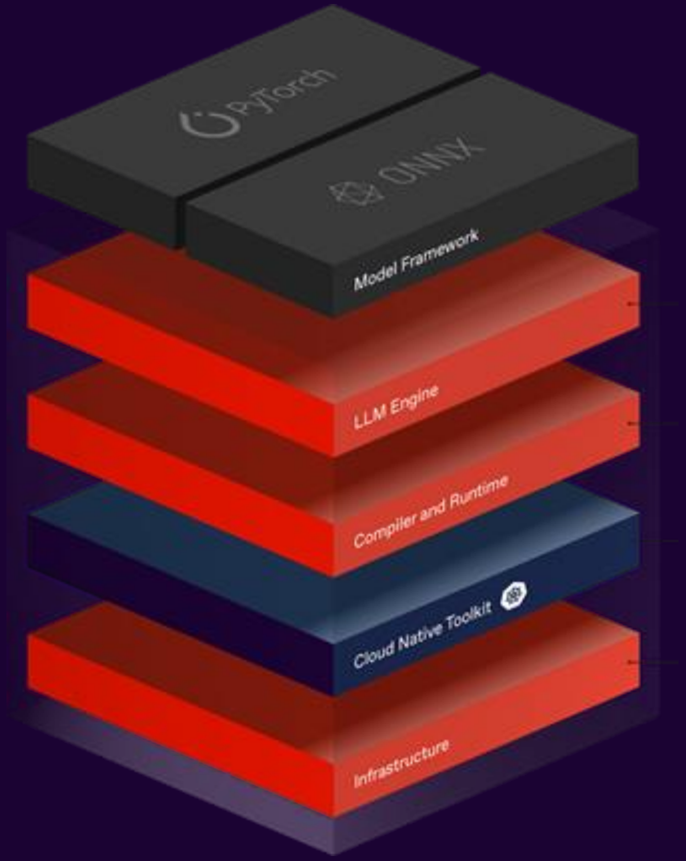




Model Execution

Serving

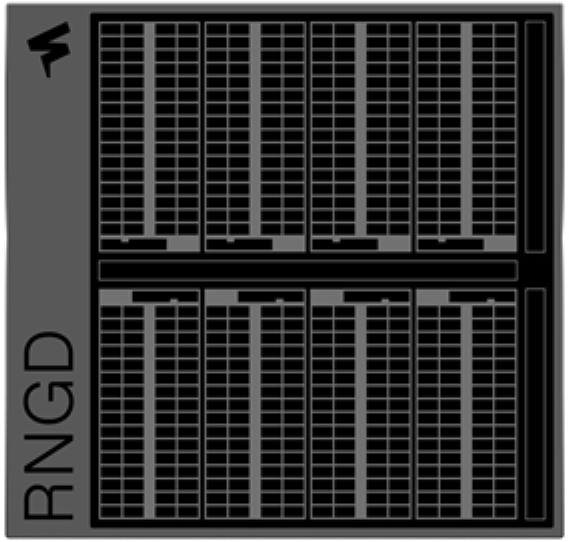
Utilization



Spatial Partitioning for Container and VM environment

A single RNGD has **8 Processing Elements (PEs)**

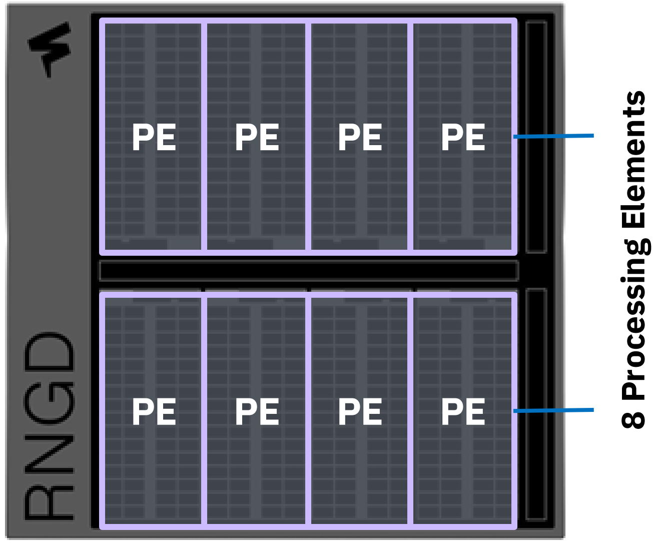
An RNGD can be spatially partitioned into many **individual NPUs**



Spatial Partitioning for Container and VM environment

A single RNGD has **8 Processing Elements (PEs)**

An RNGD can be spatially partitioned into many **individual NPUs**

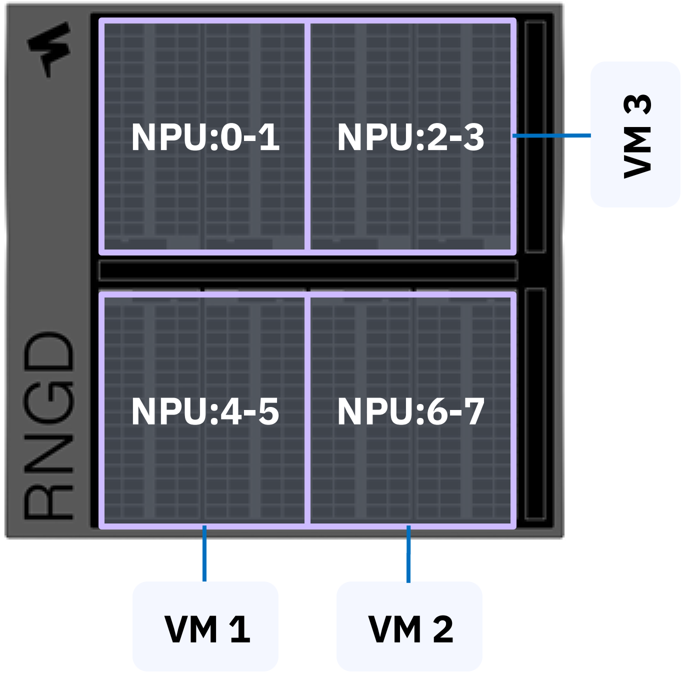


Spatial Partitioning for Container and VM environment

A single RNGD has **8 Processing Elements (PEs)**

An RNGD can be spatially partitioned into many **individual NPUs**

Up to 4 PEs can operate together as **a single NPU**

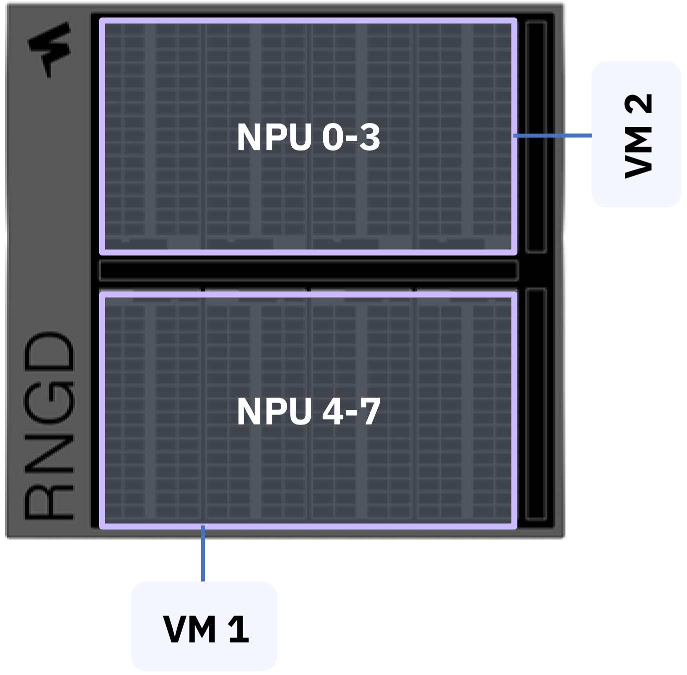


Spatial Partitioning for Container and VM environment

A single RNGD has **8 Processing Elements (PEs)**

An RNGD can be spatially partitioned into many **individual NPUs**

Up to 4 PEs can operate together as **a single NPU**



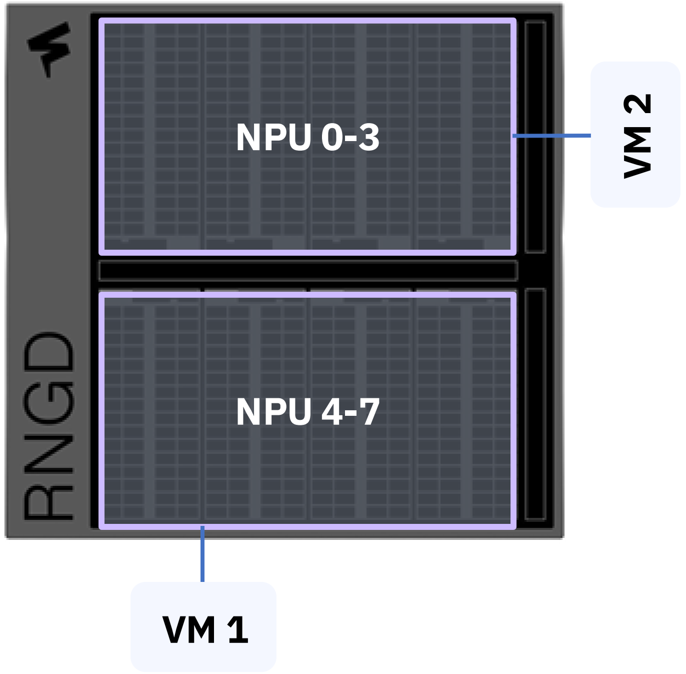
Spatial Partitioning for Container and VM environment

A single RNGD has **8 Processing Elements (PEs)**

An RNGD can be spatially partitioned into many **individual NPUs**

Up to 4 PEs can operate together as **a single NPU**

Furiosa RNGD supports **SR-IOV** (Single Root IO Virtualization) **for multiple isolated access from VMs**



Furiosa Software Stack Key Features

PyTorch 2.0 integration

Quantization toolkit (FP8, INT8, INT4, ..)

3D model parallelism support

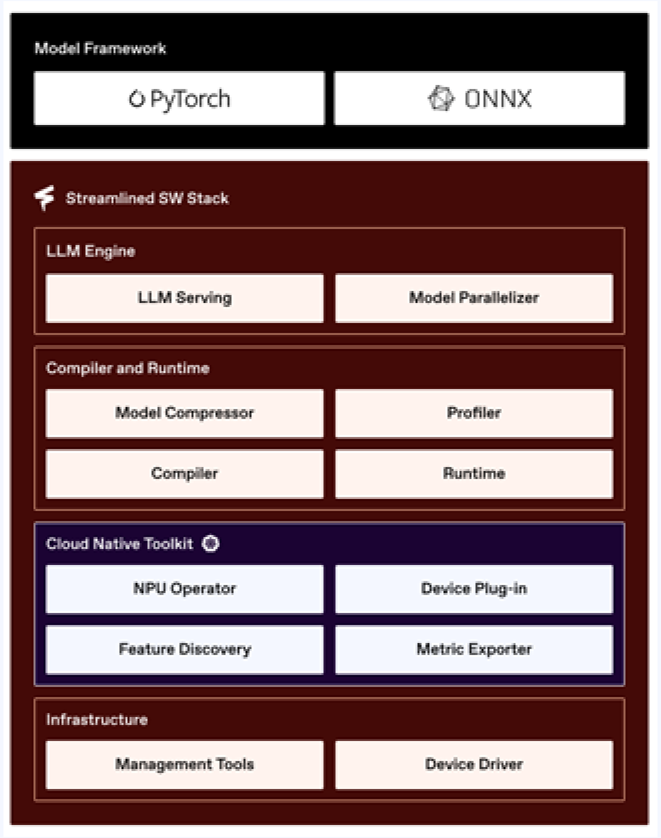
Graph compiler for DNN models

Performance profiling tools

LLM serving framework compatible with vLLM

Kubernetes device plugin and NPU operator

Virtual machine support



In summary

Delivering peak AI performance with high efficiency requires

Maximized model efficiency

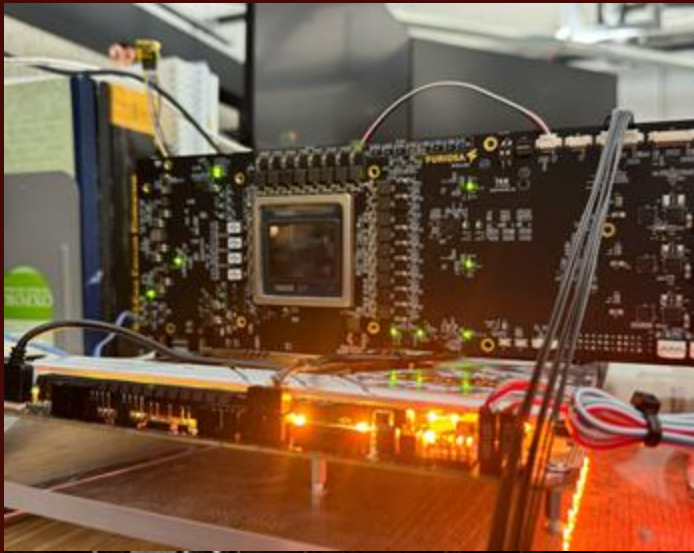
The RNGD Chip, Compiler, and Furiosa Quantizer deliver peak performance with low-precision inference for speed and efficiency.

Enhanced serving capabilities

Boost throughput and reduce latency in production with PagedAttention, Blocked KV cache, and continuous batching.

Flexible resource utilization

RNGD's spatial partitioning and SR-IOV ensure optimal resource allocation, maximizing NPU utilization in virtualized and containerized environments.



In order to solve for mass AI adoption,
We have to think beyond *just* hardware