



# Groq's approach to HW/SW Systems for LLM Inference.

A deep dive

Hello. |



**Sanjif Shanmugavelu**  
Software Engineer, Groq  
[sshanmugavelu@groq.com](mailto:sshanmugavelu@groq.com)

#### Acknowledgements

Valentin Reis, Igor Arsovski, Andrew Bitar, Tobias Becker,  
Paul Brown, Dan Gard, Dinesh Maheshwari, Santosh  
Raghavan, Sarit Schube, Bill Xing, Peter Lillian, **and all the  
others.**

We're at a tipping point

## Training ➔ Inference

McKinsey  
& Company

“AI inferencing hardware alone in the data center will be **2x that for AI training** hardware by 2025.”

McKinsey &  
Company



“In our trailing four quarters, we estimate that inference drove about 40% of our Data Center revenue.”

Jensen Huang, NVIDIA  
Q1 2025 Earnings

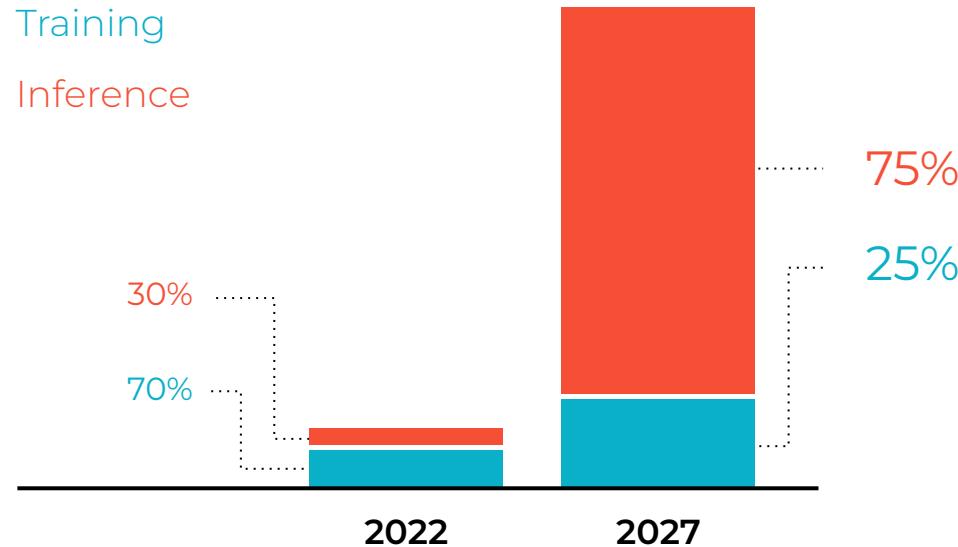
We're at a tipping point

## Training ➔ Inference

AI Semiconductors Deployed in Data Centers

Training

Inference



Gartner, Forecast Analysis AI Semiconductors Worldwide, April 2023

# Challenges

What do we care about as a community?

## Training

vs

## Inference

### Run Time

Weeks or months

Milliseconds or seconds

### Challenges

TCO (Cost, Energy)

TCO (Cost, Energy)

**Speed** (LLMs: Per user **Token Rates**)

### Model Size

- Parameter Volume
- LLMs: Context Length

# Fast AI Inference.

V1 LPU-based systems compared to leading GPU architectures in the market

↓**4x**

Cost

↓**3x**

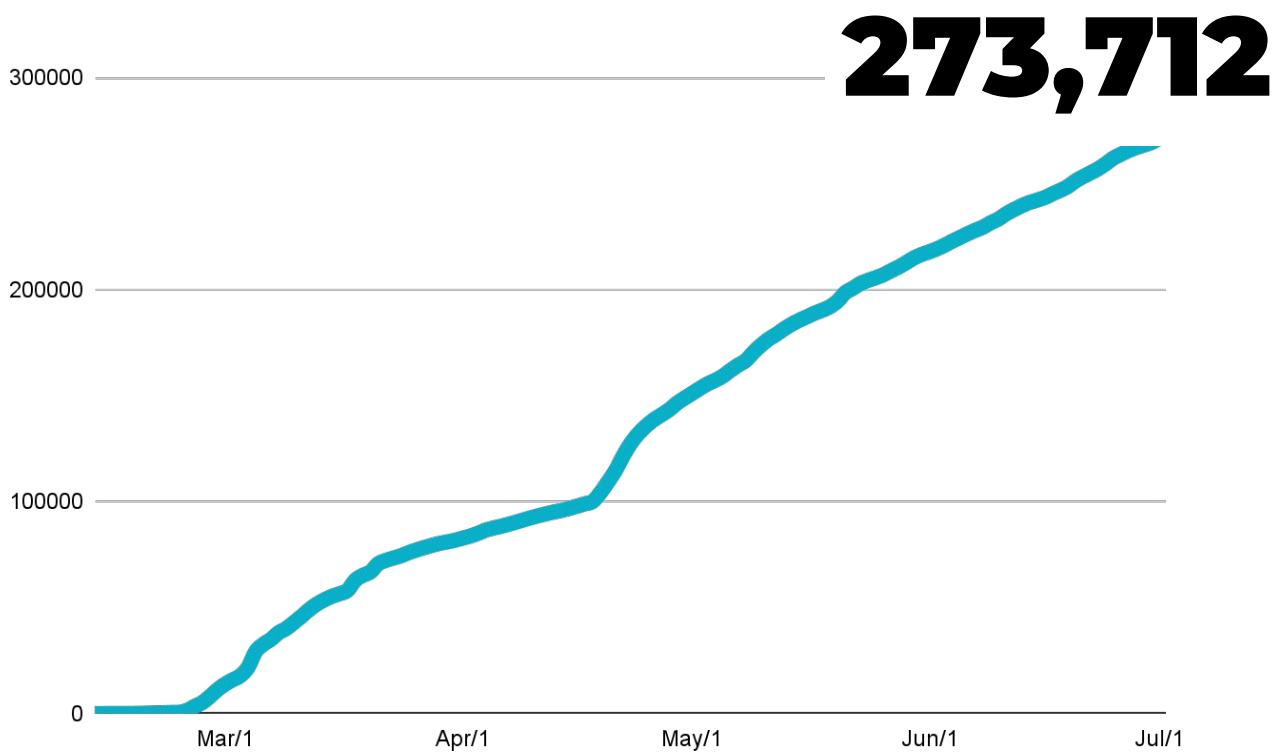
Energy

↑**6x**

Speed

Cumulative growth

Developers



Cumulative growth

Active API Keys

150000

**133,528**

100000

50000

0

Mar/1

Apr/1

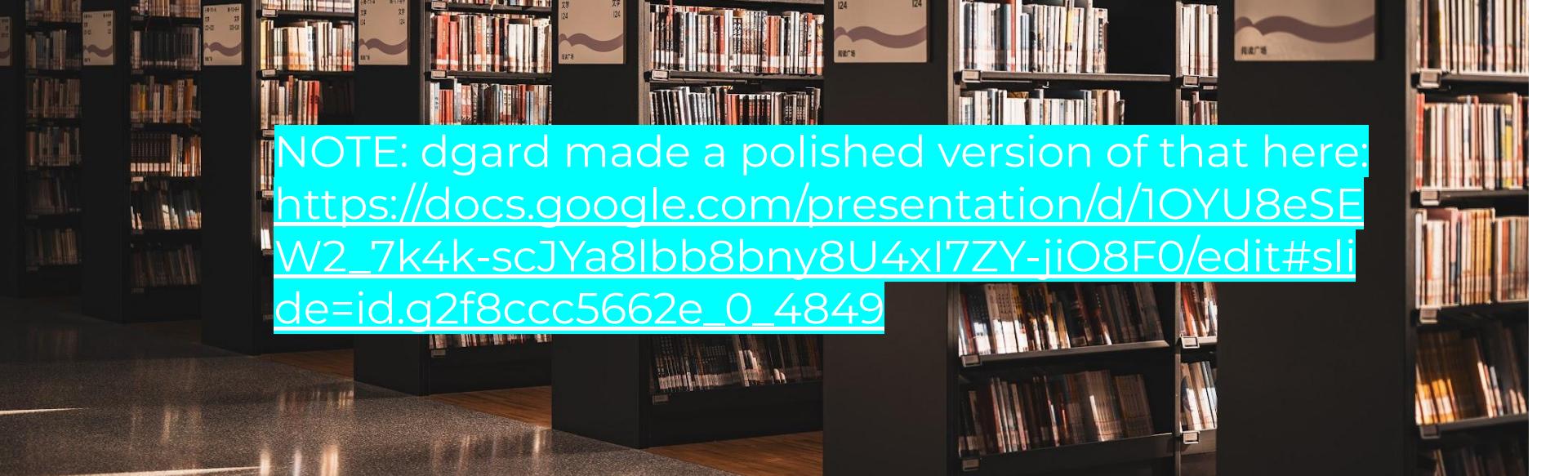
May/1

Jun/1

Jul/1

# Agenda

- 1. LLM Inference is GEMM/GEMV**
- 2. GPU Scaling Challenges**
- 3. Language Processing Units (LPUs)**
- 4. Scaling Inference on LPUs**



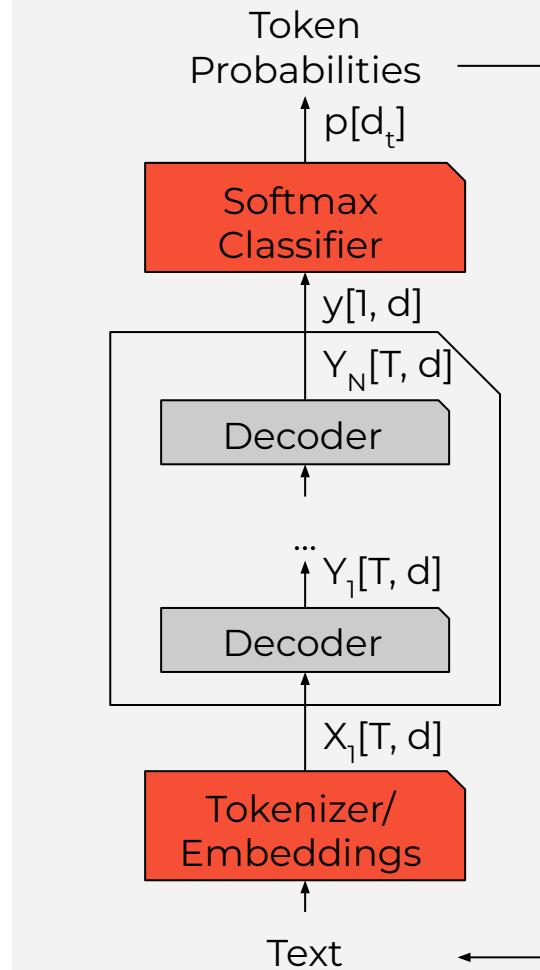
NOTE: dgard made a polished version of that here:  
[https://docs.google.com/presentation/d/1OYU8eSEW2\\_7k4k-scJYa8lbb8bny8U4xl7ZY-jiO8F0/edit#slide=id.g2f8ccc5662e\\_0\\_4849](https://docs.google.com/presentation/d/1OYU8eSEW2_7k4k-scJYa8lbb8bny8U4xl7ZY-jiO8F0/edit#slide=id.g2f8ccc5662e_0_4849)

# LLM Inference is GEMM/GEMV

↑ Input Processing, Output Generation and **KV Caches!**

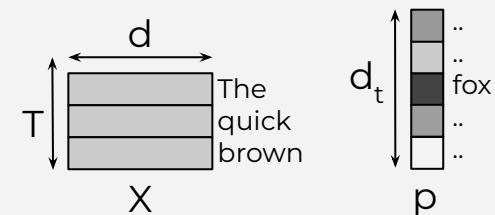
# Architecture

Sketch of a causal decoder-based LM



d: embedding dimension  
d<sub>t</sub>: token dictionary size  
T: sequence length  
N: layers

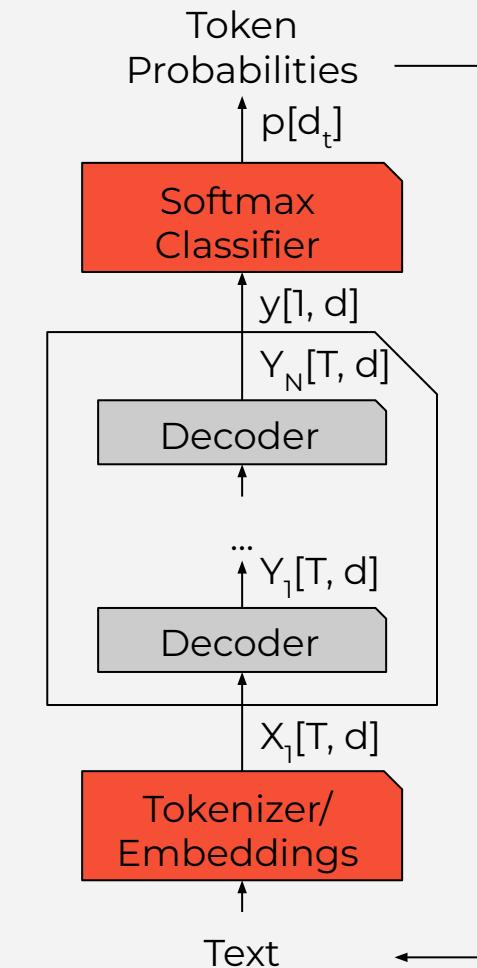
Not shown: Layer normalization, positional encoding, etc.



# Architecture

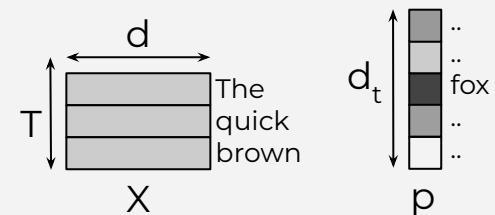
Sketch of a causal decoder-based LM

- **Decoders** [1] are the workhorse



d: embedding dimension  
d<sub>t</sub>: token dictionary size  
T: sequence length  
N: layers

Not shown: Layer normalization, positional encoding, etc.

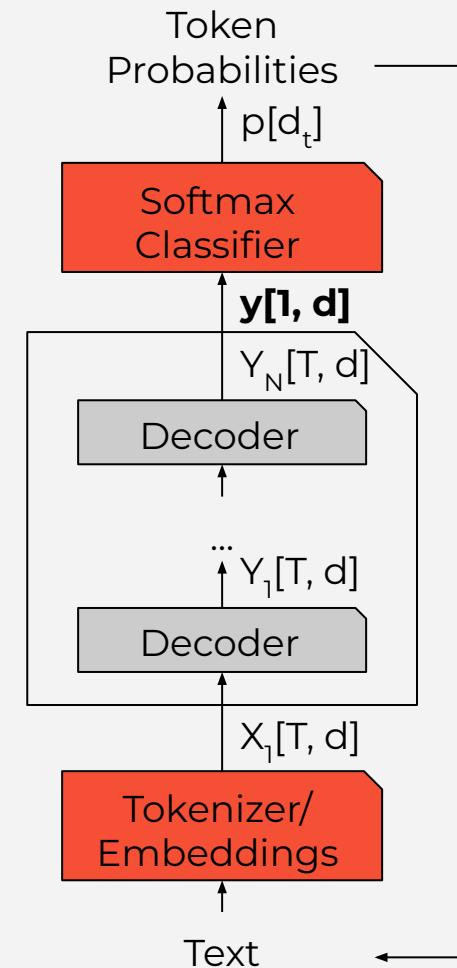


[1]: Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

# Architecture

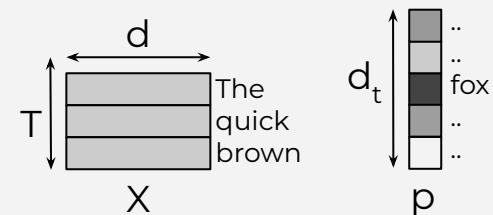
Sketch of a causal decoder-based LM

- **Decoders** [1] are the workhorse
- **Softmax Classifier** and **Embeddings** are position independent



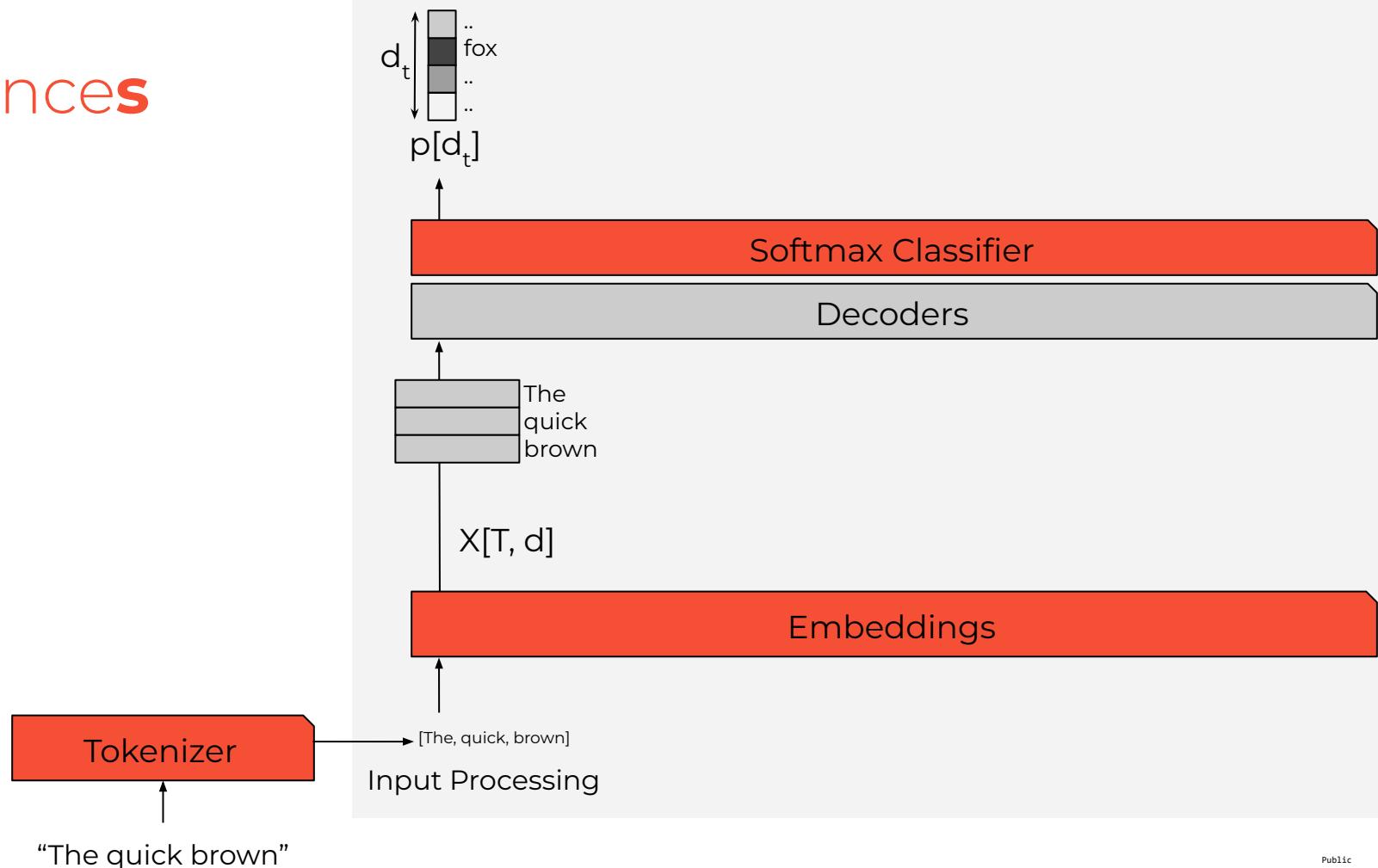
d: embedding dimension  
 $d_t$ : token dictionary size  
T: sequence length  
N: layers

Not shown: Layer normalization, positional encoding, etc.

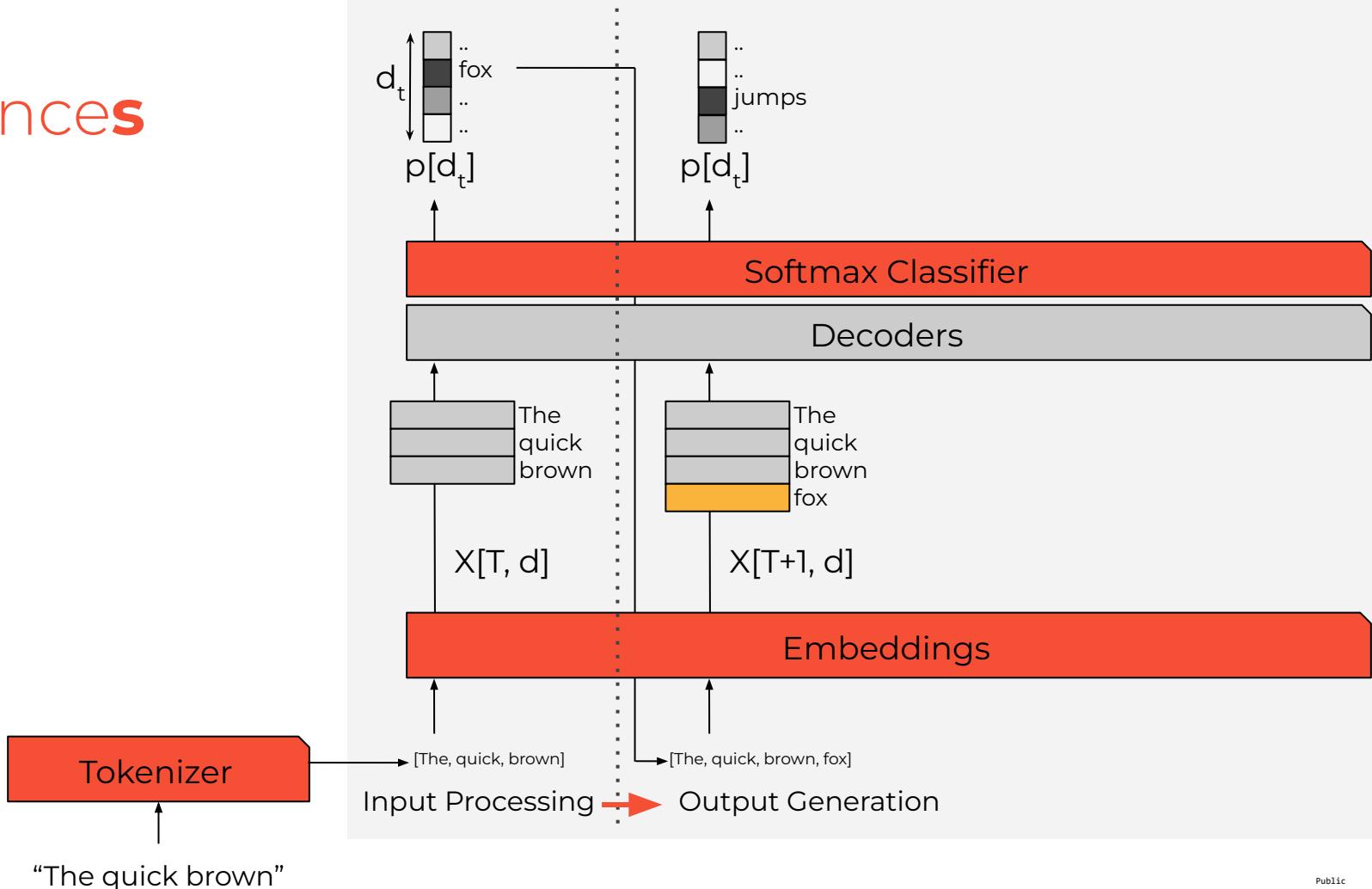


[1]: Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

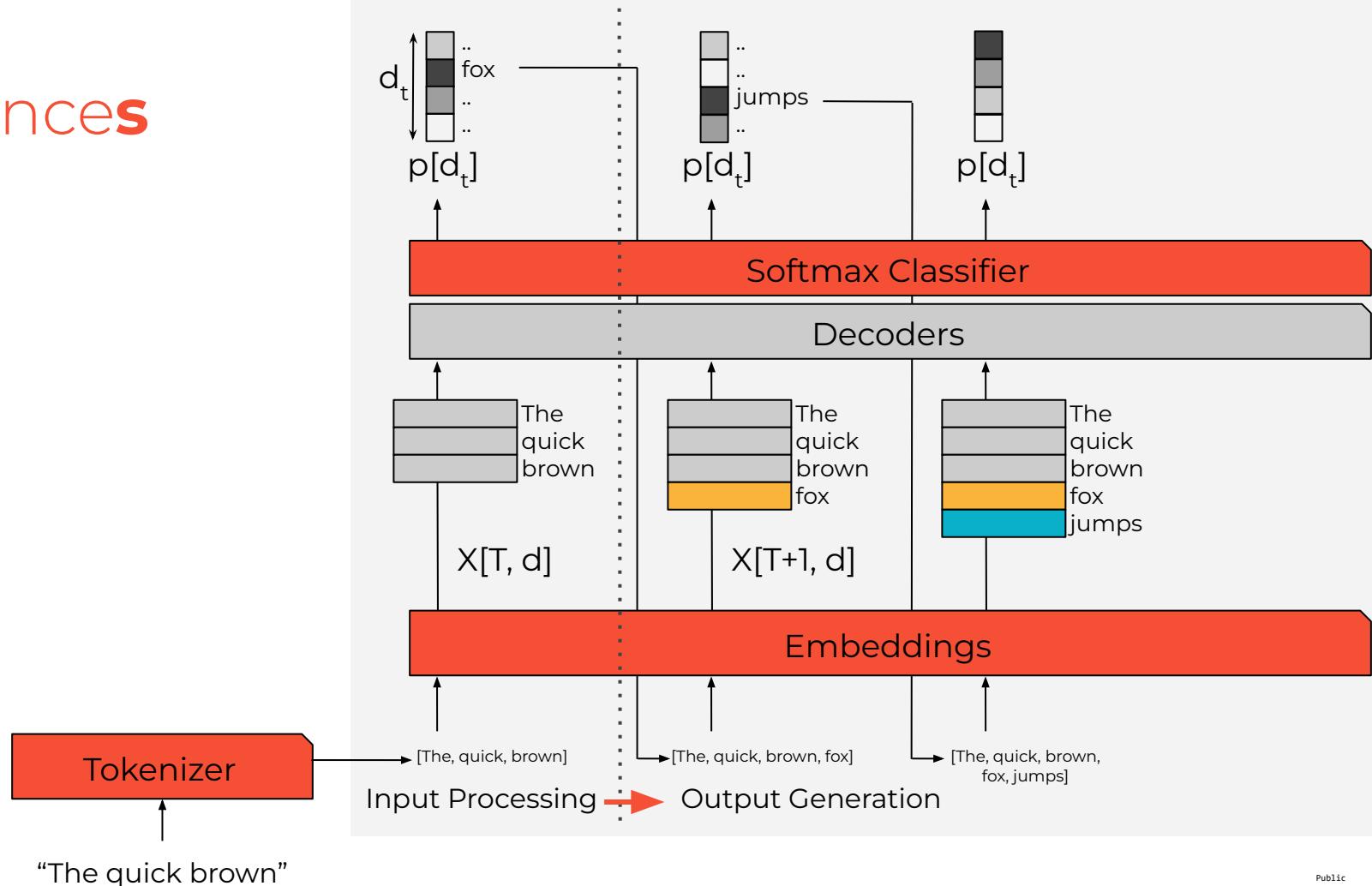
# Inferences



# Inferences

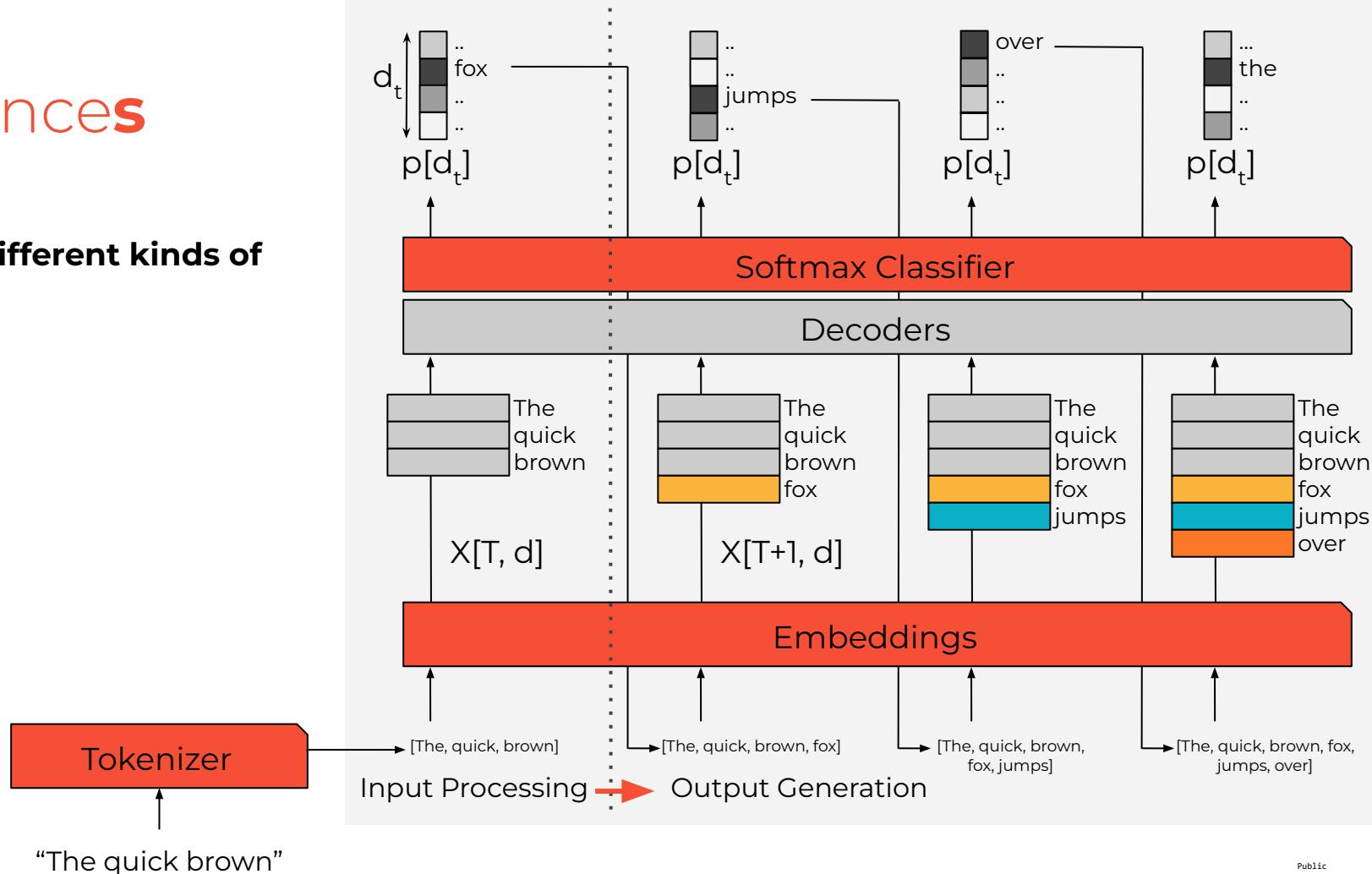


# Inferences



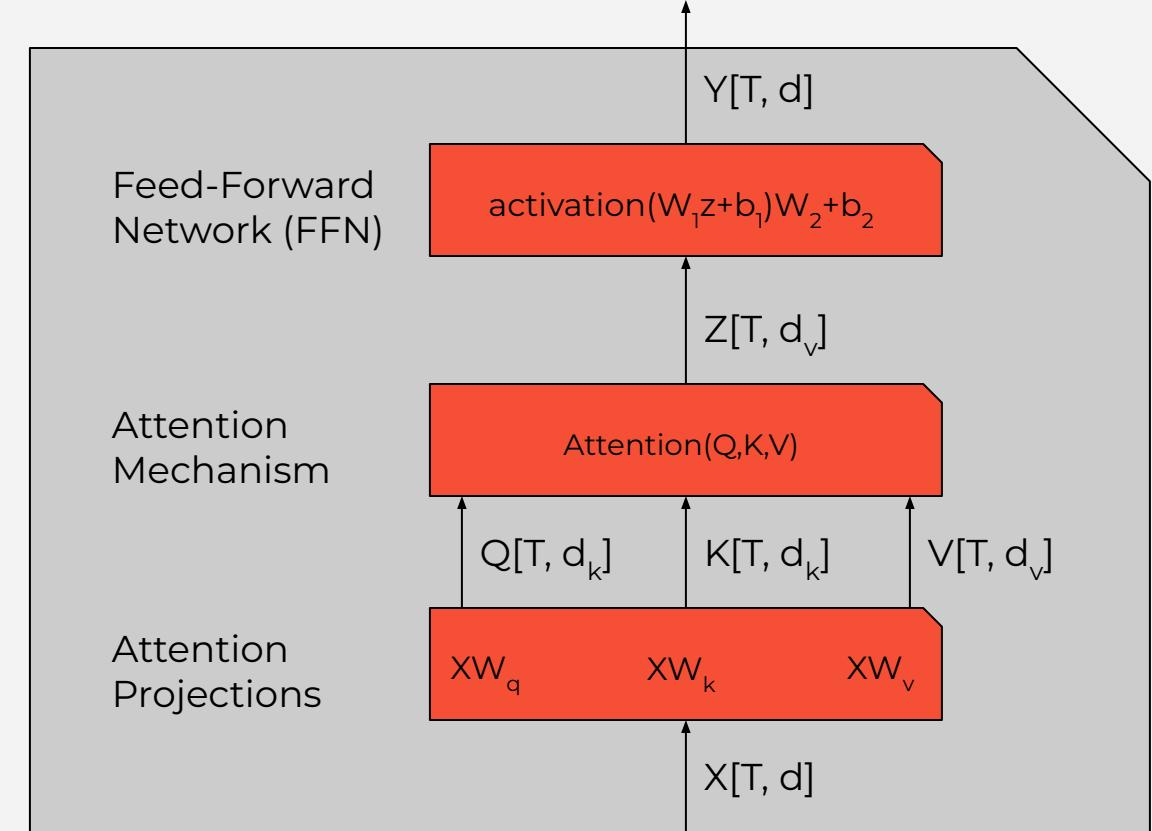
# Inferences

- Two different kinds of tasks.



# Decoders

The workhorse of LLM inference

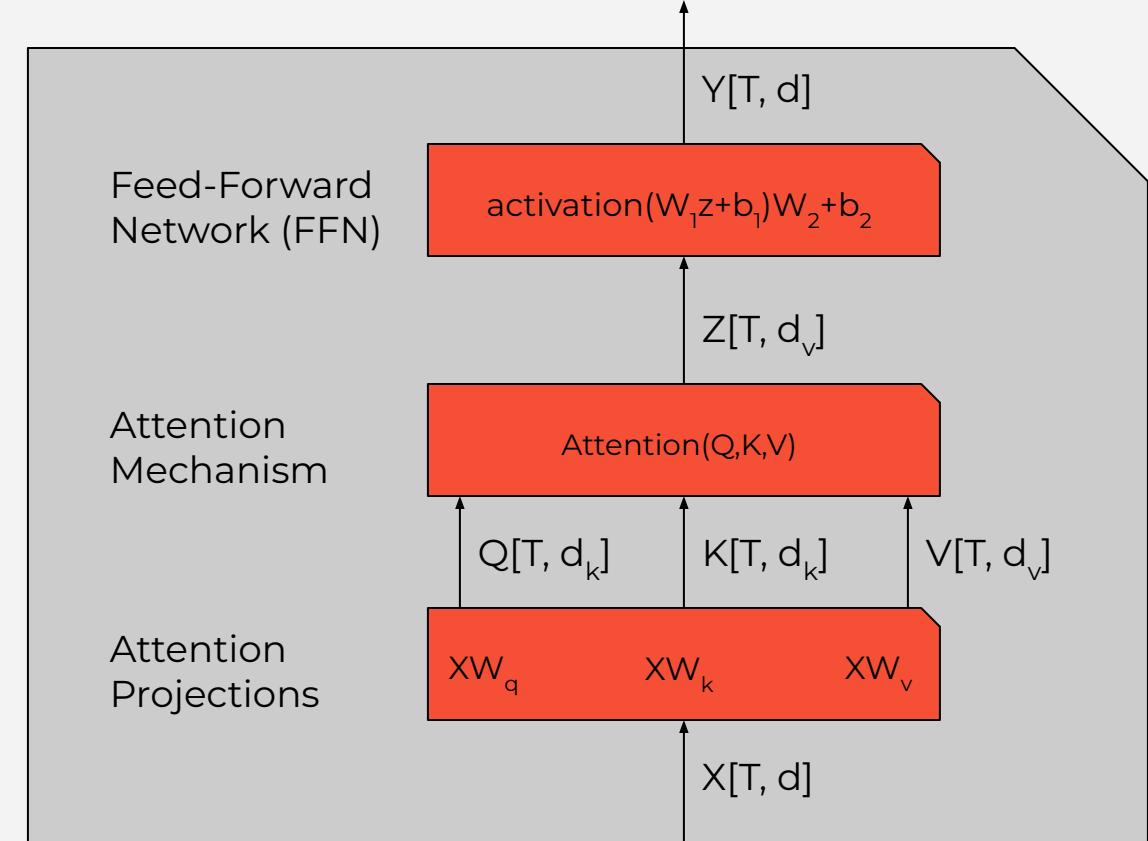


Simplifications: No layer normalization, rotary positional embeddings, mixture of experts, multi-head/multi-query/grouped-query attention..

# Decoders

The workhorse of LLM inference

## ■ BLAS 2

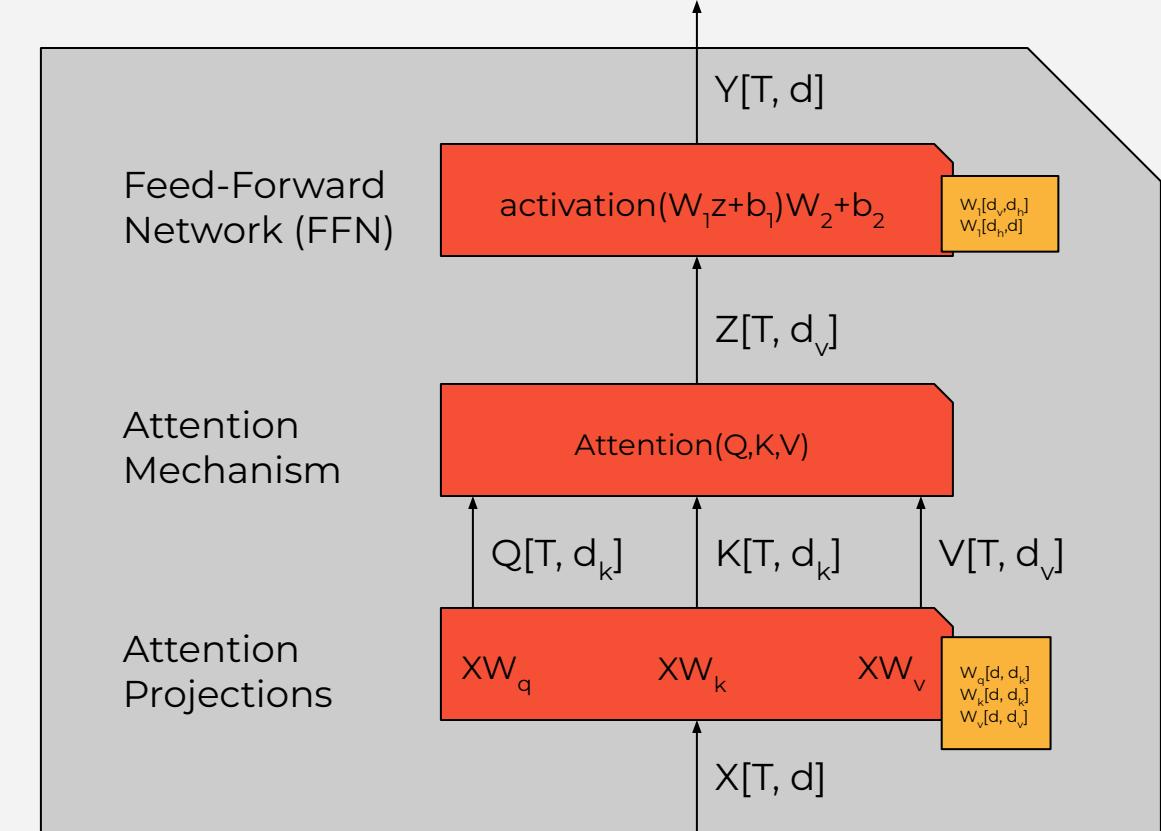


Simplifications: No layer normalization, rotary positional embeddings, mixture of experts, multi-head/multi-query/grouped-query attention..

# Decoders

The workhorse of LLM inference

- **BLAS 2**
- Position independent FFN/Projections



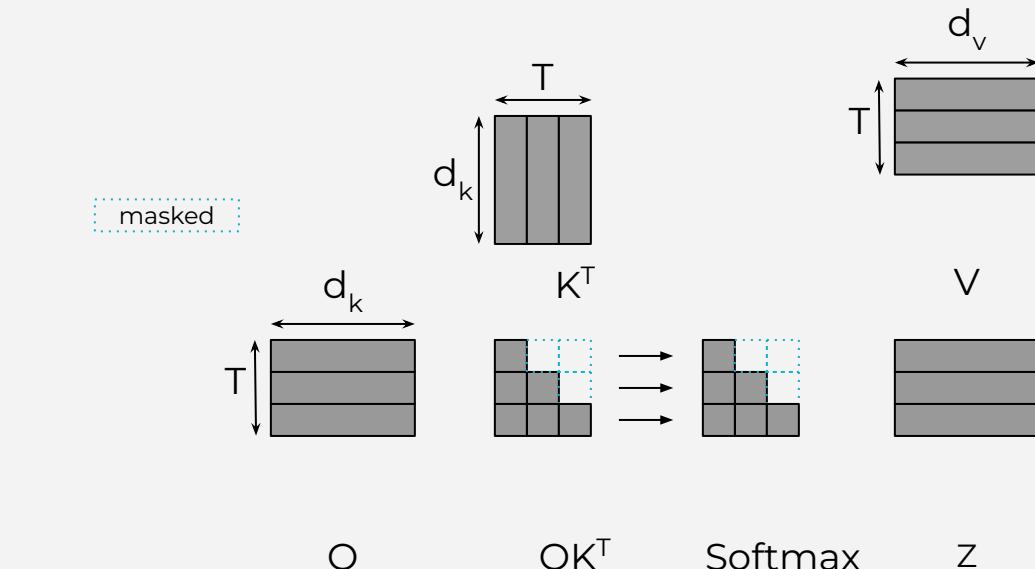
Simplifications: No layer normalization, rotary positional embeddings, mixture of experts, multi-head/multi-query/grouped-query..

# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$

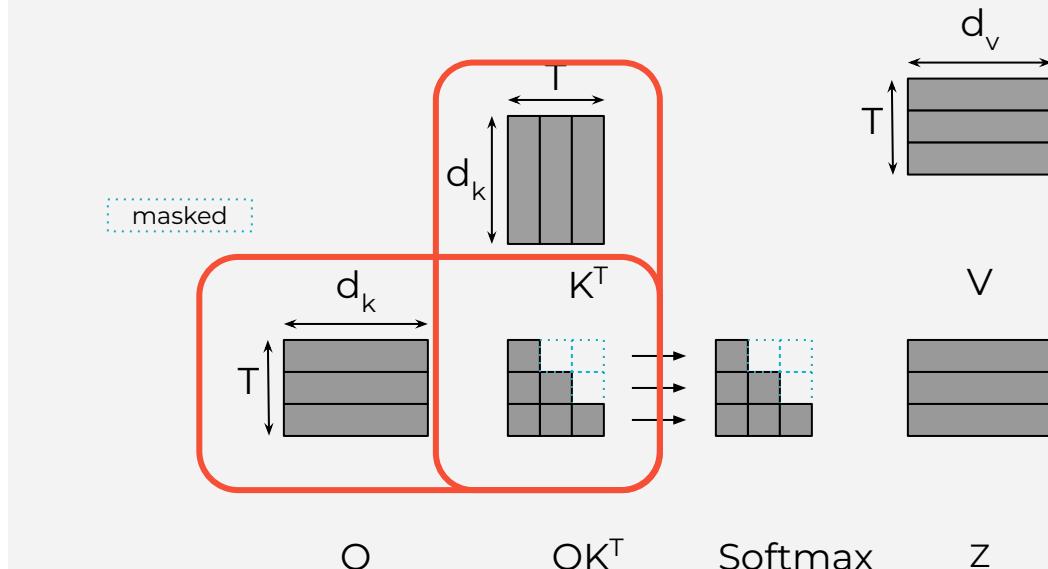


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(\underline{QK^T})}{\sqrt{d_k}} \right) V$$

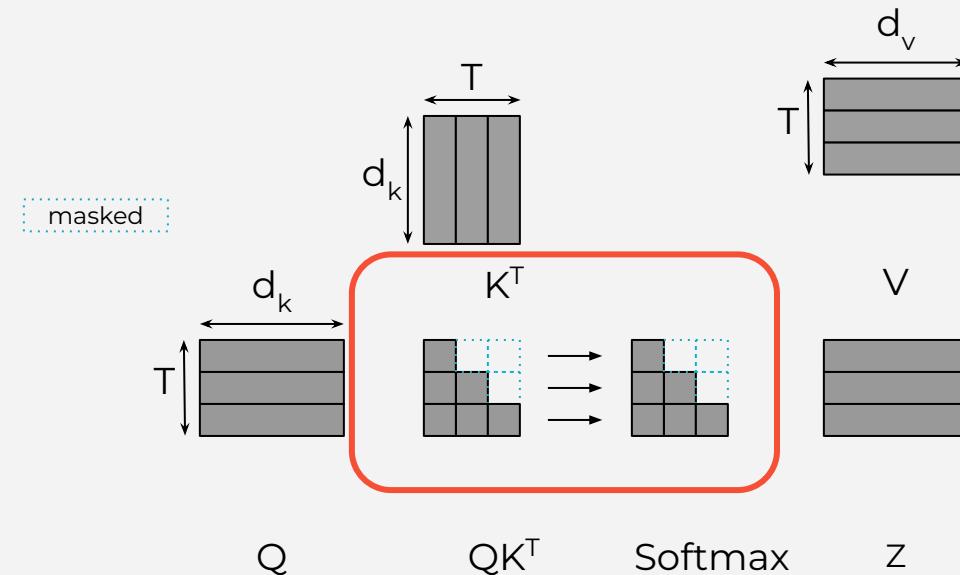


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \underline{\text{Softmax}} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$

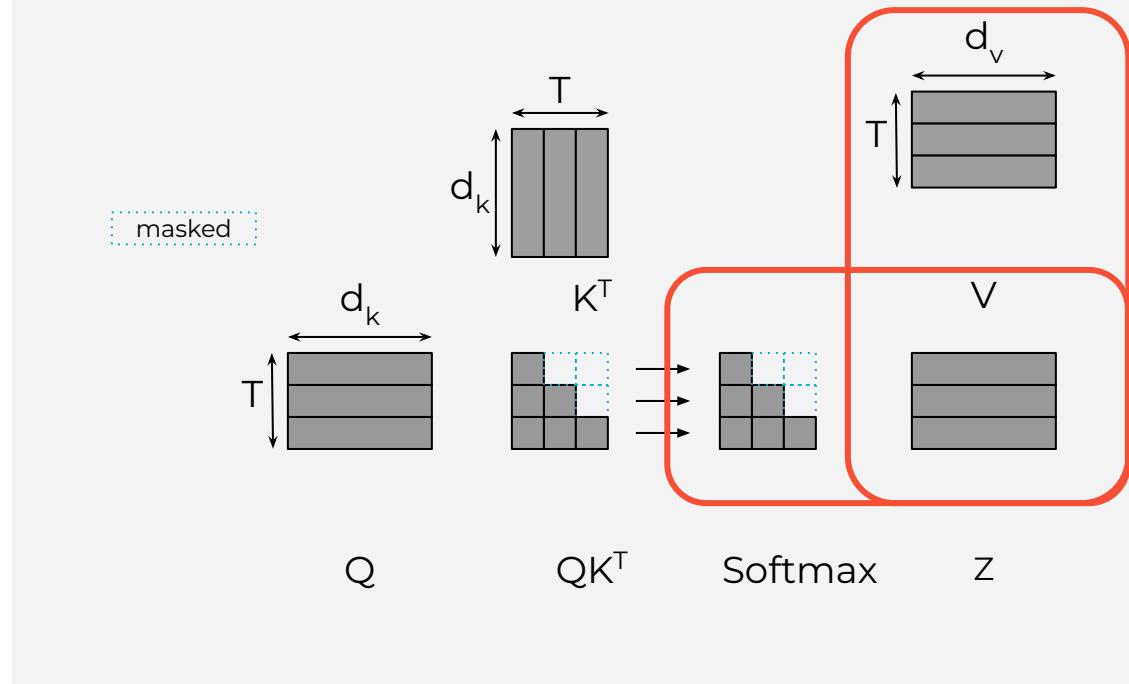


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$

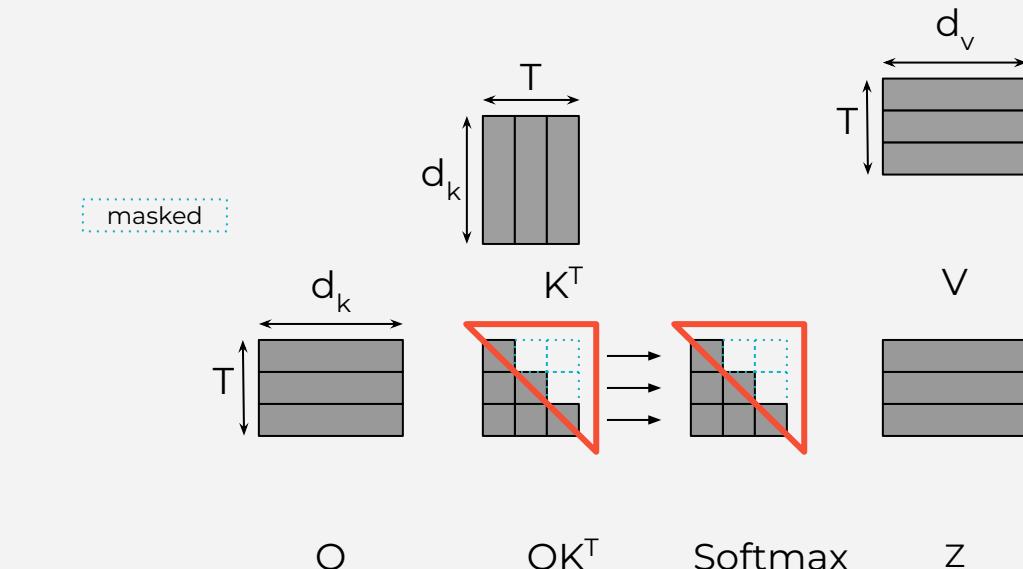


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$

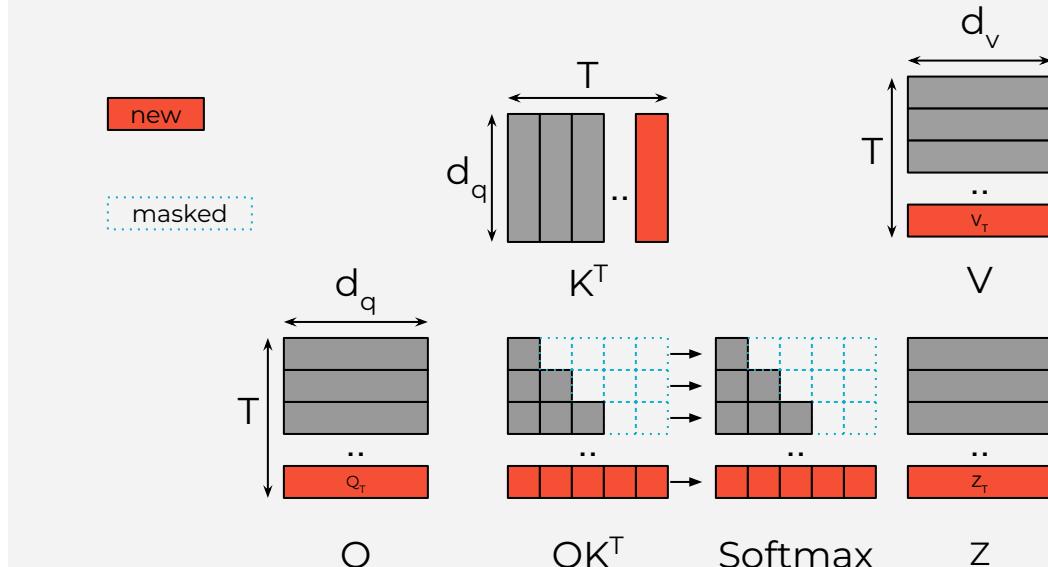


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens

$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$

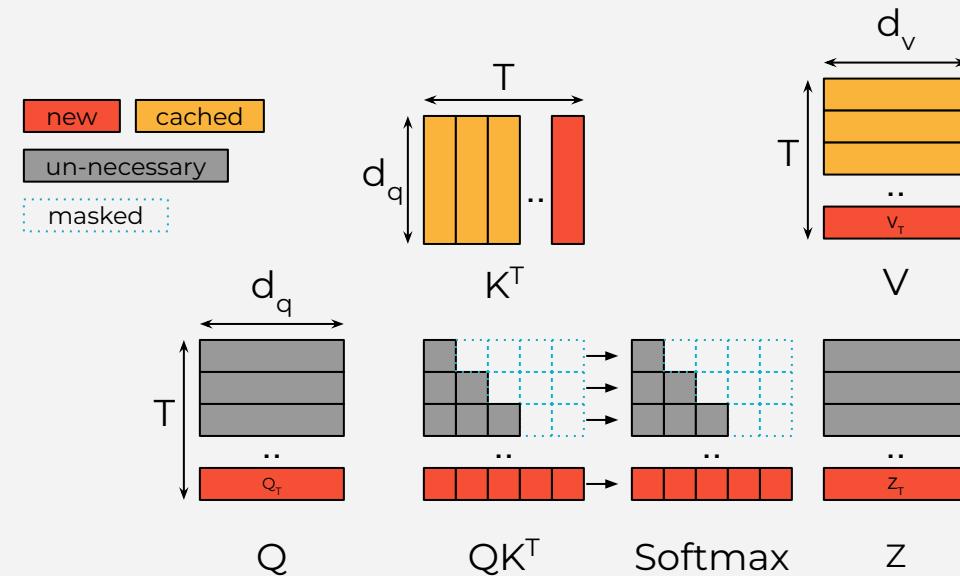


# Attention

Position dependent, but memoizable

- **Causal** attention **only** depends on previous tokens
- **KV Caching** makes output generation **GEMV**

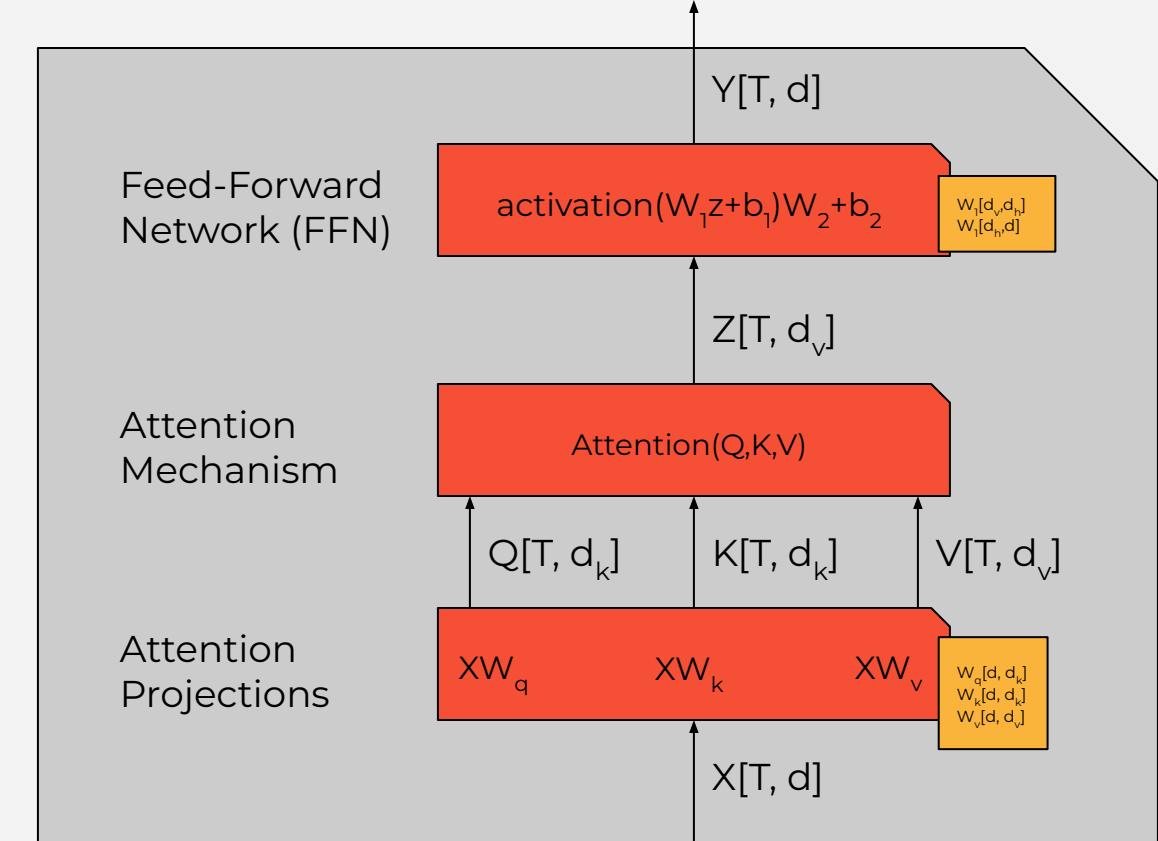
$$\text{Attention}(Q, K, V) = \text{Softmax} \left( \frac{\text{mask}(QK^T)}{\sqrt{d_k}} \right) V$$



# Decoders

The workhorse of LLM inference

- Position independent FFN/Projections
- **BLAS 2**

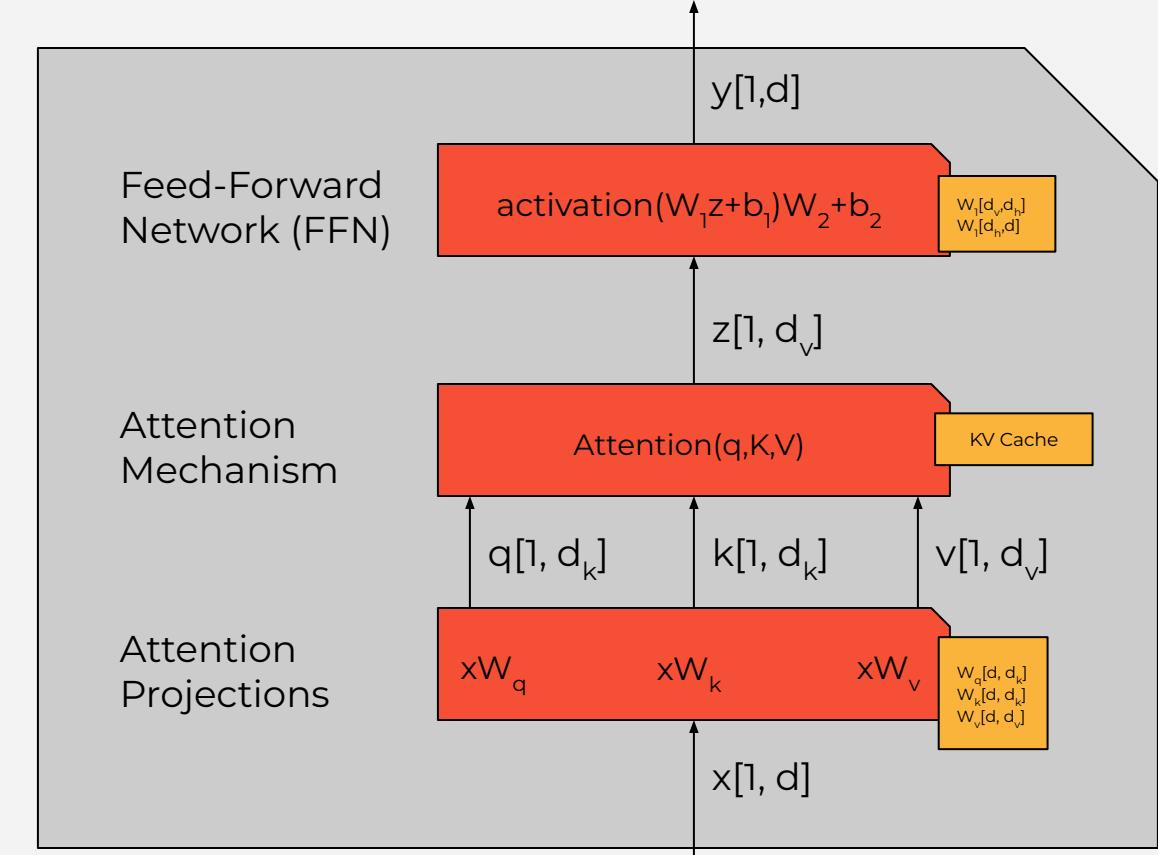


Simplifications: No layer normalization, rotary positional embeddings, mixture of experts, multi-head/multi-query/grouped-query attention..

# Decoders

The workhorse of LLM inference

- Position independent FFN/Projections
- **BLAS 2**
- **GEMM** input processing (compute bound)
- **GEMV** output generation (memory bound)



Simplifications: No layer normalization, rotary positional embeddings, mixture of experts, multi-head/multi-query/grouped-query attention..

# Getting Larger

As of July 2024..

- **Performance** and **Calibration** improve with model size [2]
- **Distributed** workload

Trained, size unknown	Claude OPUS, GPT4, Gemini, Inflection, Olympus
Training, size known	<i>AuroraGPT 1T (ANL)</i> <i>Llama3 400B (Meta)</i>
Trained, size known	GPT3 175B (OpenAI) PaLM 540B (Google) <i>Falcon 180B (TII)</i> <i>LLAMA3 70B (Meta)</i>

Some popular LLMs and their parameter sizes

[2]: Srivastava, Aarohi, et al. "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models." *arXiv preprint arXiv:2206.04615* (2022).

# Memory Reqs

LLAMA3-70B

BF16, 8-way Grouped Query Attention (GQA) [3]

- Weights: **140GB**
- KV Caches: **2.68GB**

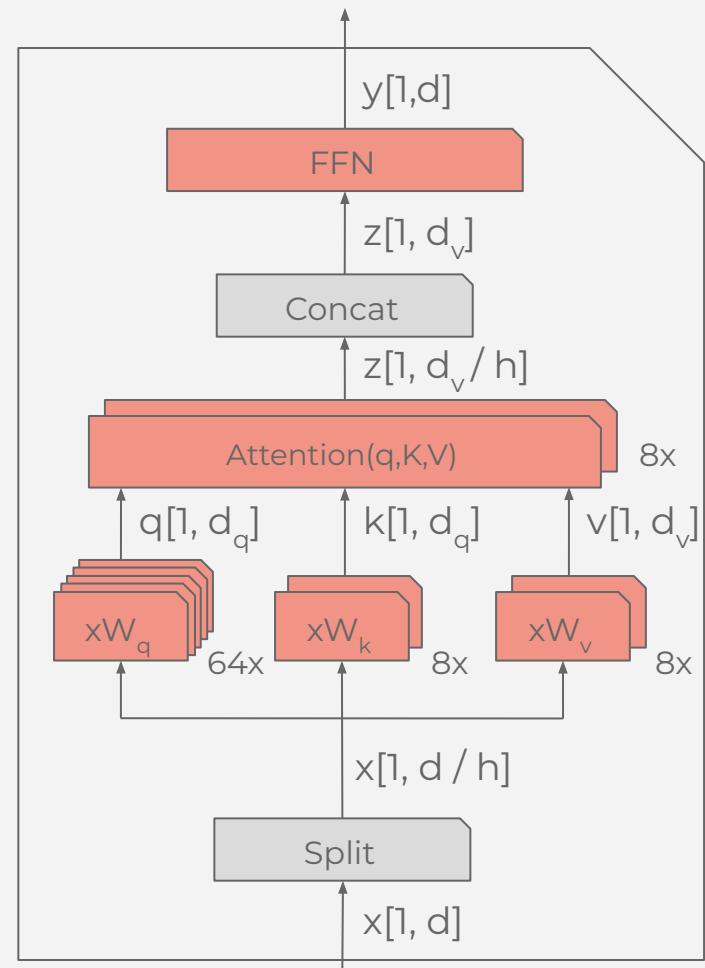
G (# KV heads)	8
h (# Q heads)	64
N (# layers)	80
d (embedding)	8192
$d_q = d_k = d_v = d/h$	128
T (ctx length)	8192

Model parameters for Int8-LlaMA3-70B

Extrapolating to 1T Model:

- Weights: **X TBs**
- KV Caches: **XX GBs**

KV Cache size =  
**numerics** \*  $d_q$  \*  $T$  Matrix size  
\* 2 K and V  
\* **N** # decoder layers  
**/G** due to G-way GQA



[3]: Ainslie, Joshua, et al. "Gqa: Training generalized multi-query transformer models from multi-head checkpoints." *arXiv preprint arXiv:2305.13245* (2023).

# Memory Reqs

LLAMA3-70B

BF16, 8-way Grouped Query Attention (GQA) [3]

- Weights: **140GB**
- KV Caches: **2.68GB**

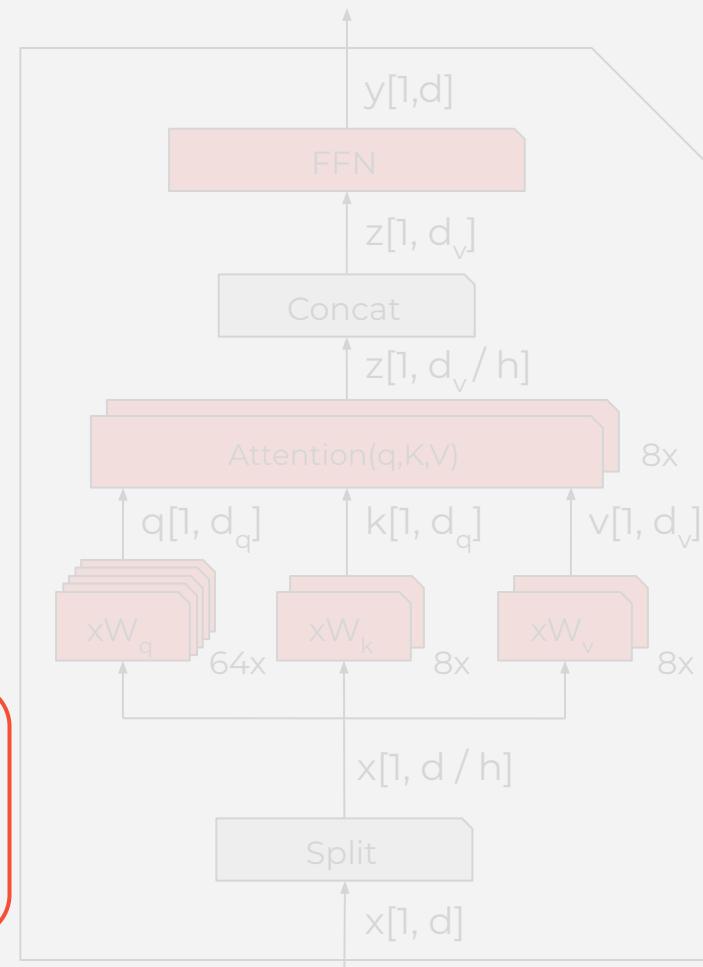
G (# KV heads)	8
h (# Q heads)	64
N (# layers)	80
d (embedding)	8192
$d_q = d_k = d_v = d/h$	128
T (ctx length)	8192

Model parameters for Int8-LlaMA3-70B

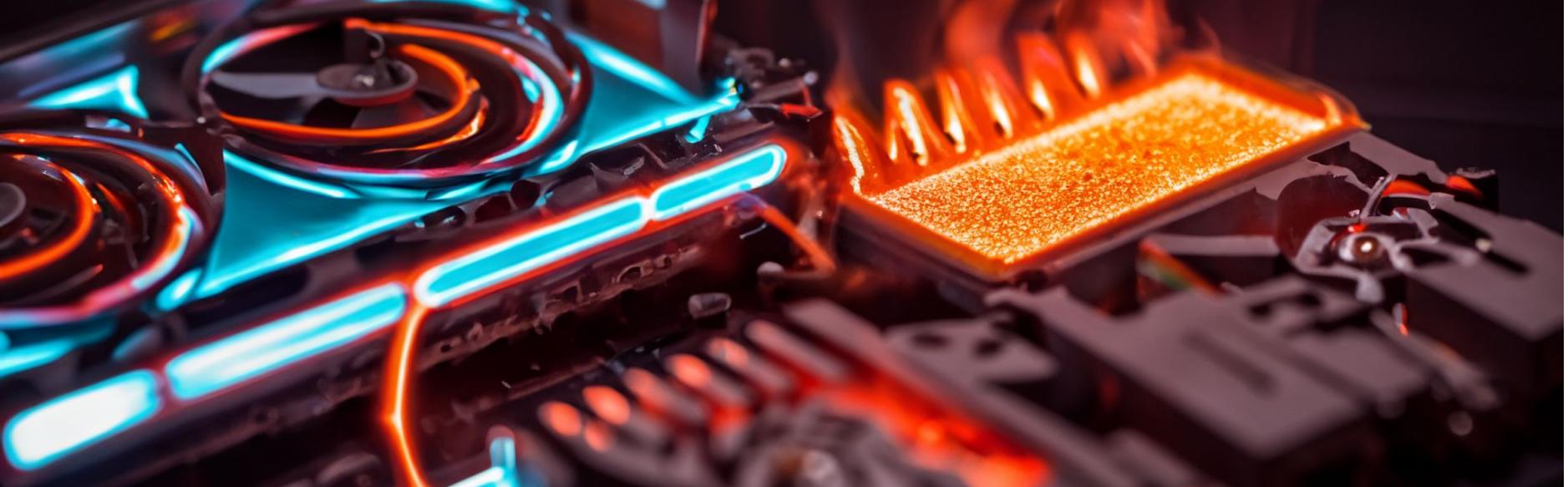
Extrapolating to 1T Model:

- Weights: **X TBs**
- KV Caches: **XX GBs**

KV Cache size =  
**numerics** \*  $d_q$  \*  $T$  Matrix size  
\* 2 K and V  
\* **N** # decoder layers  
/ **G** due to G-way GQA



[3]: Ainslie, Joshua, et al. "Gqa: Training generalized multi-query transformer models from multi-head checkpoints." *arXiv preprint arXiv:2305.13245* (2023).



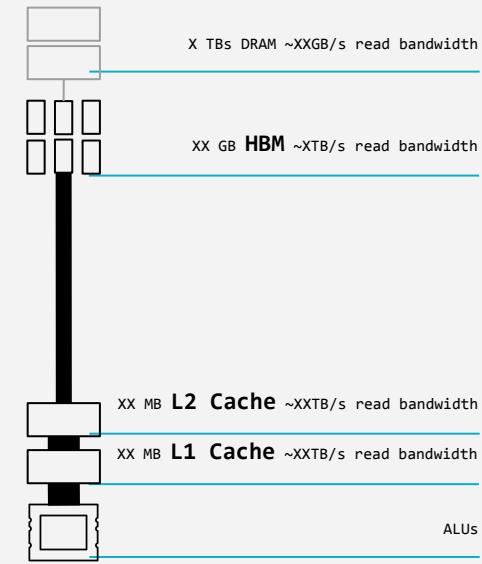
# GPUs Scaling Challenges.

↑ Scaling (mostly) in **Time, High-Batch** at output generation

# GPU Systems

Memory Hierarchy Based

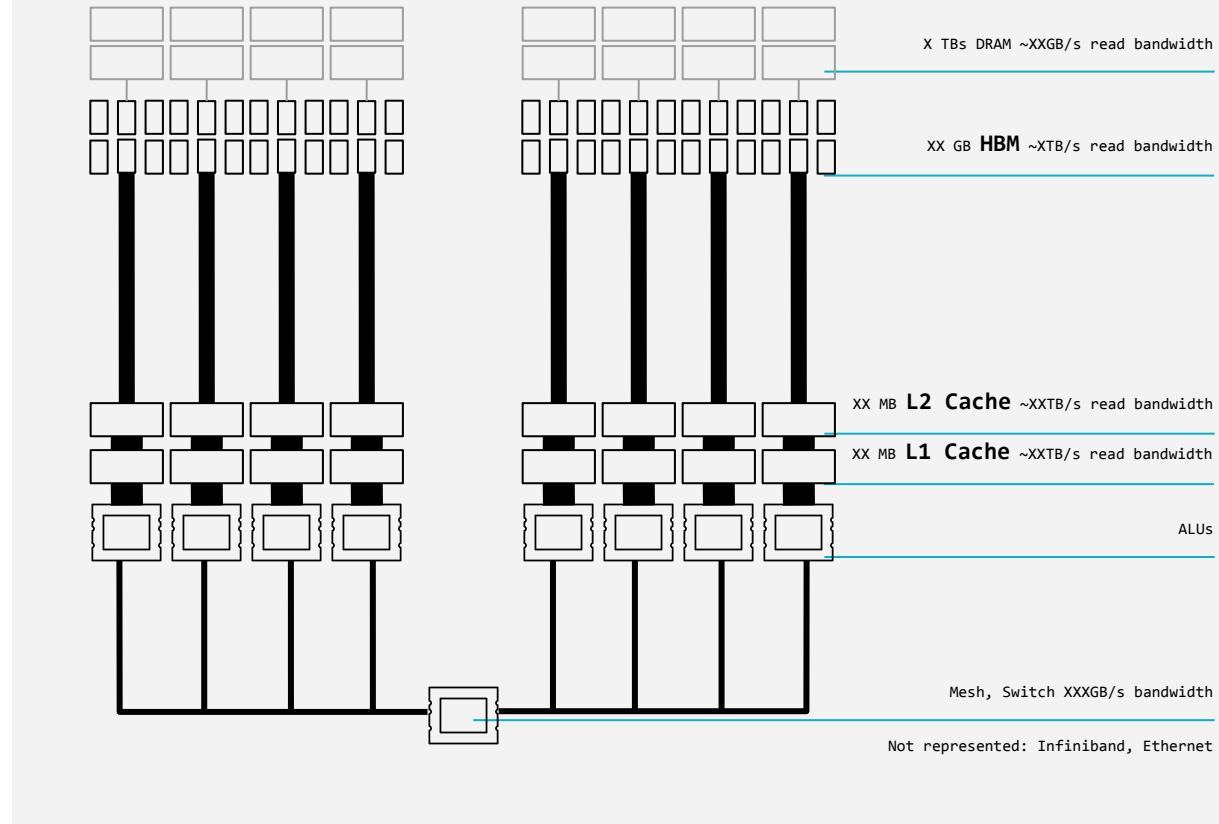
- **HBM based** memory hierarchy



# GPU Systems

Memory Hierarchy Based

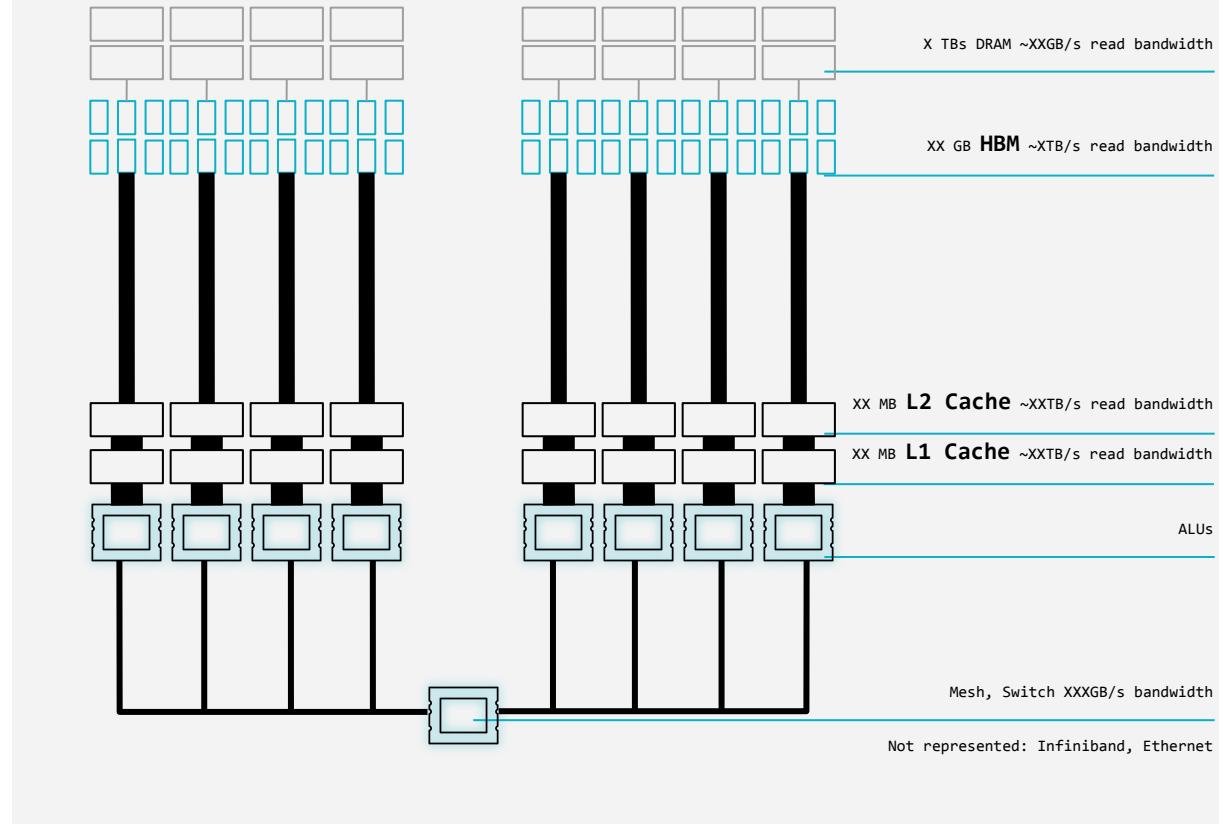
- **HBM based** memory hierarchy
- **Switch**-based networking at scale



# GPU Systems

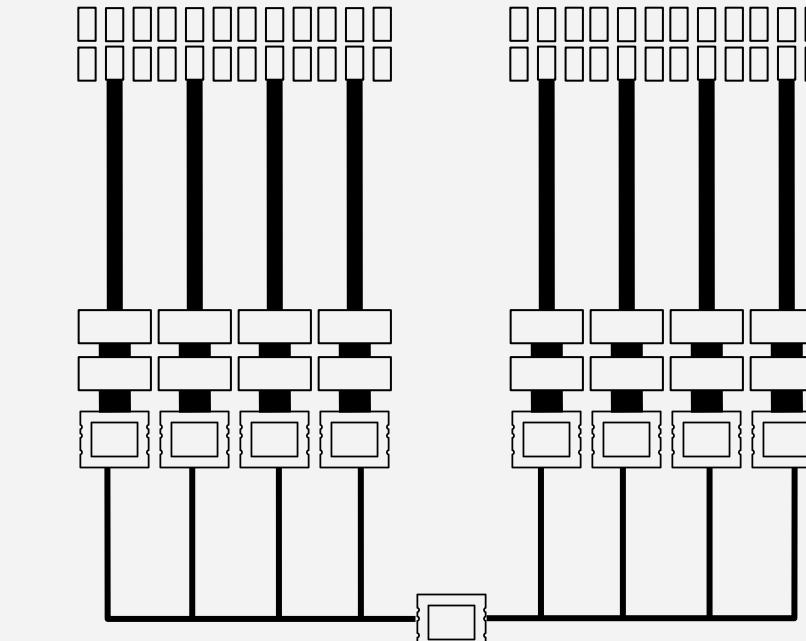
Memory Hierarchy Based

- **HBM based** memory hierarchy
- **Switch**-based networking at scale
- **Expensive BOM & Supply Concerns:** HBM, exotic packaging, switches, etc



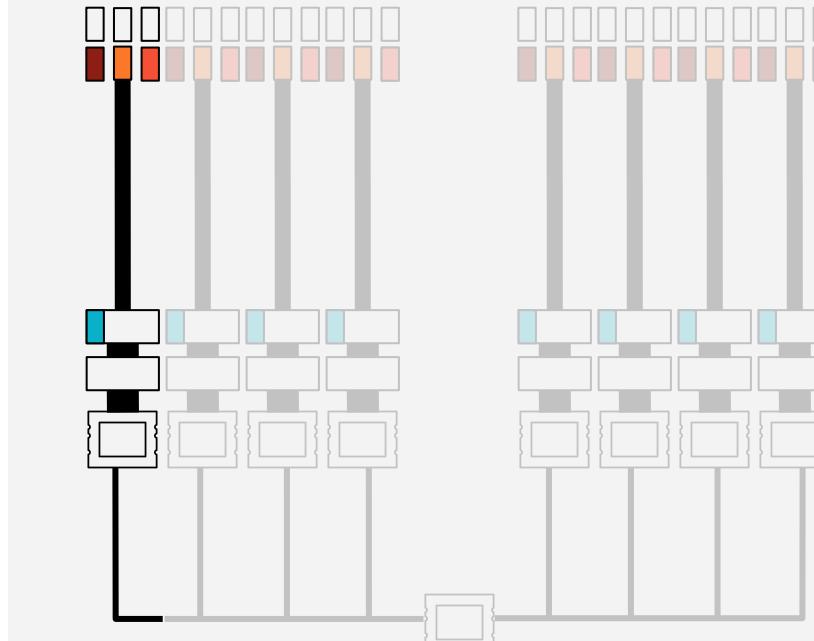
# Scaling in Time

Paging Costs



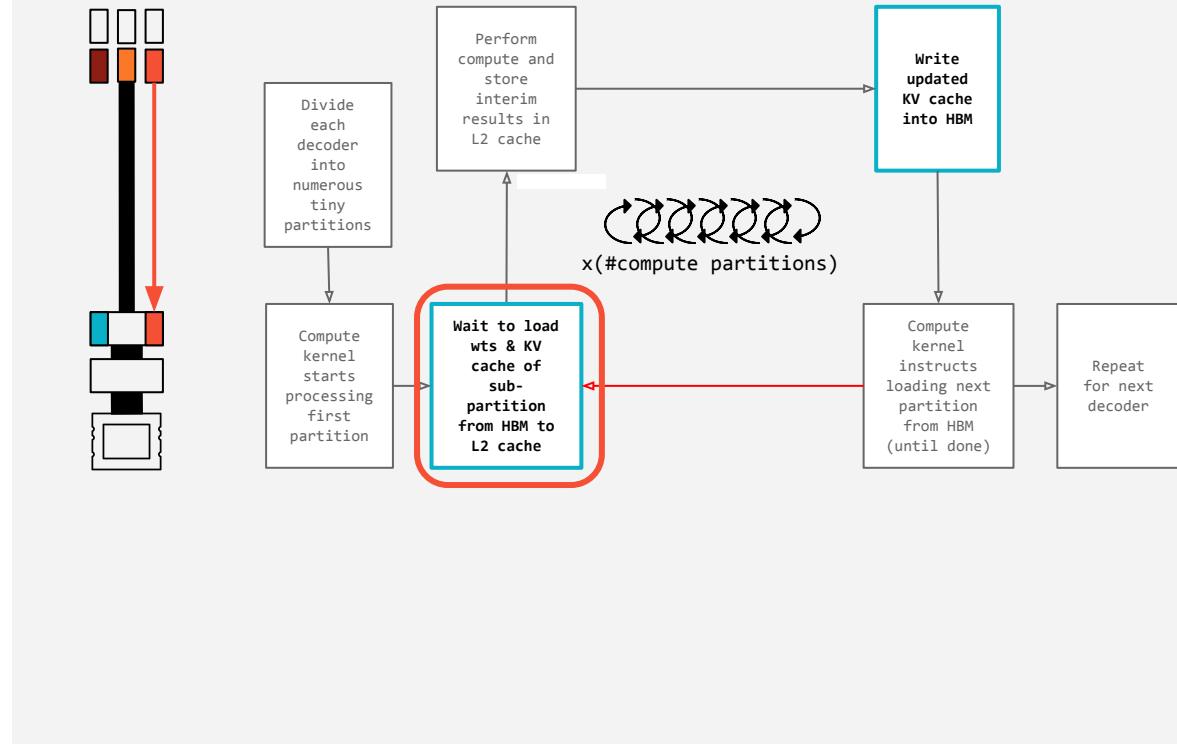
# Scaling in Time

Paging Costs



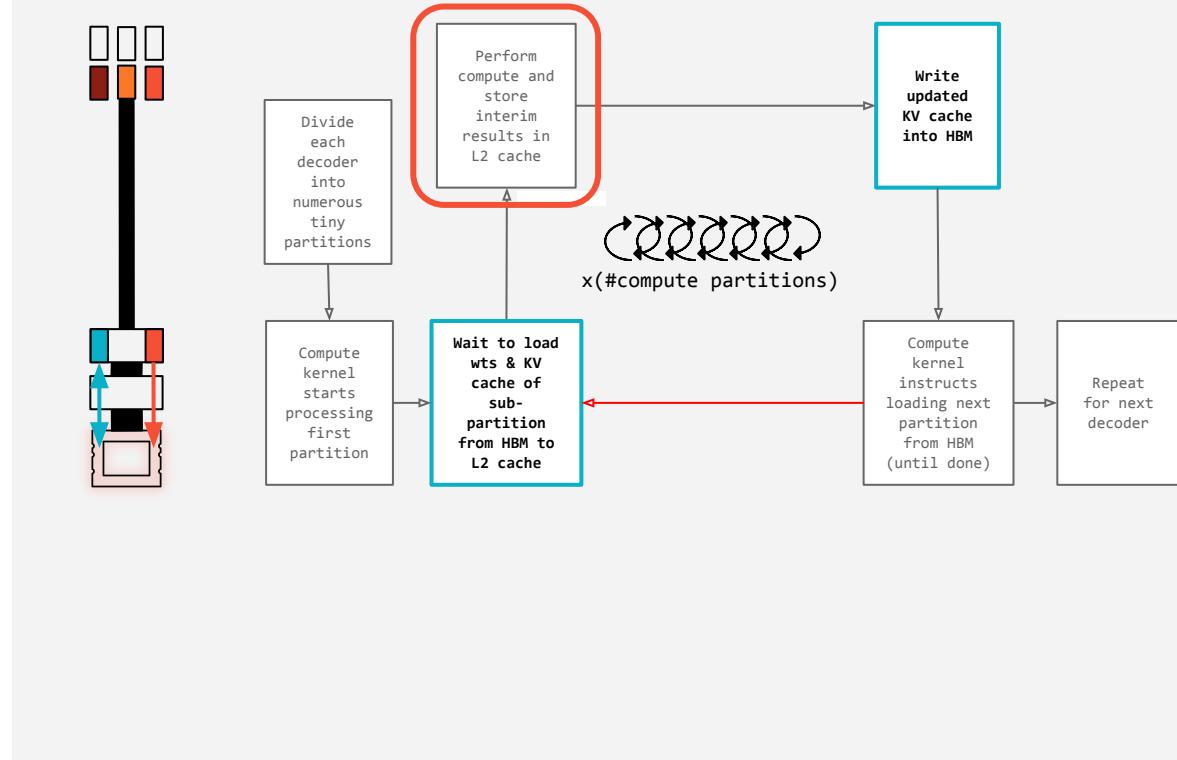
# Scaling in Time

## Paging Costs



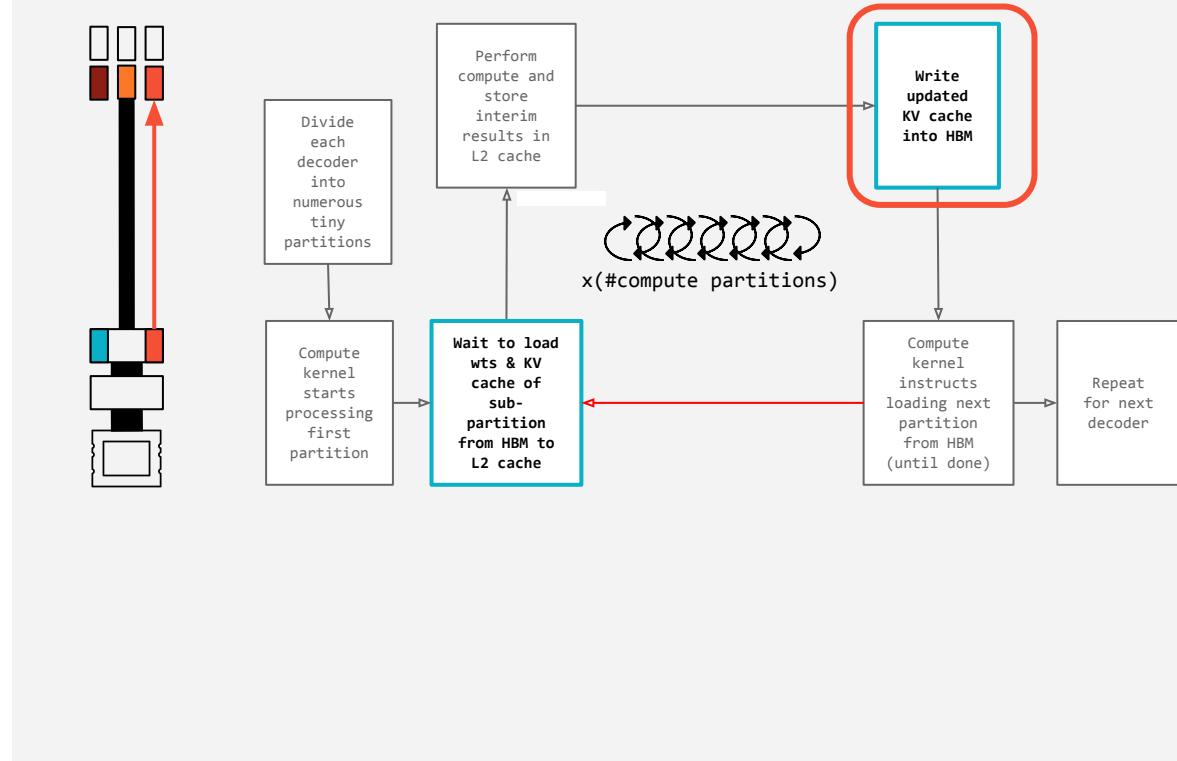
# Scaling in Time

## Paging Costs



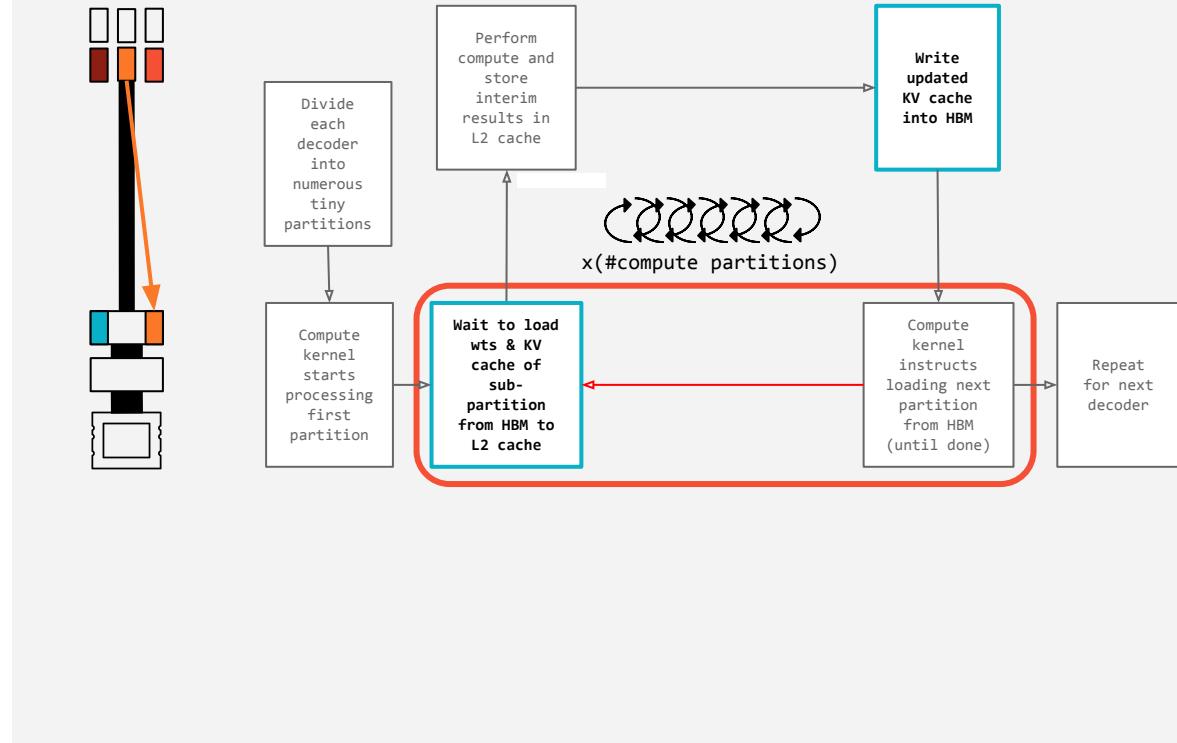
# Scaling in Time

## Paging Costs



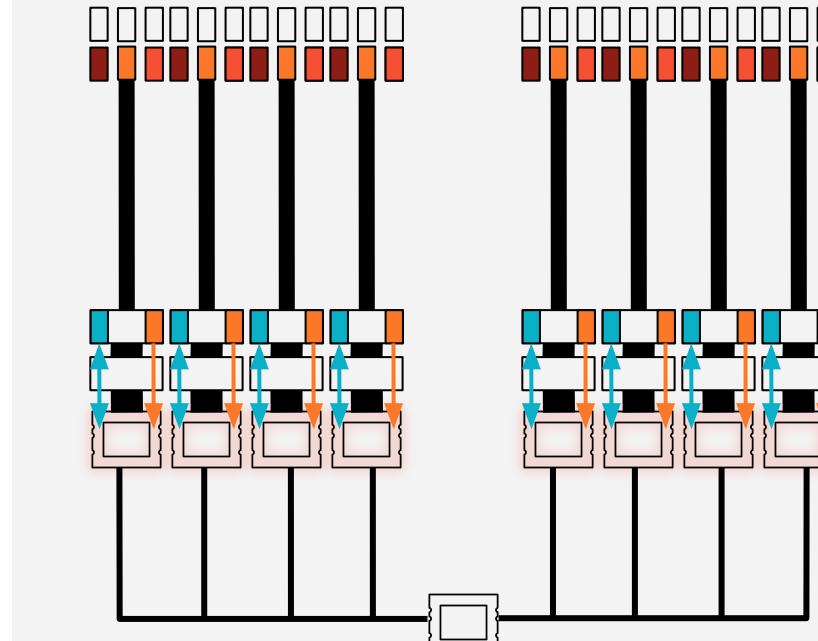
# Scaling in Time

## Paging Costs



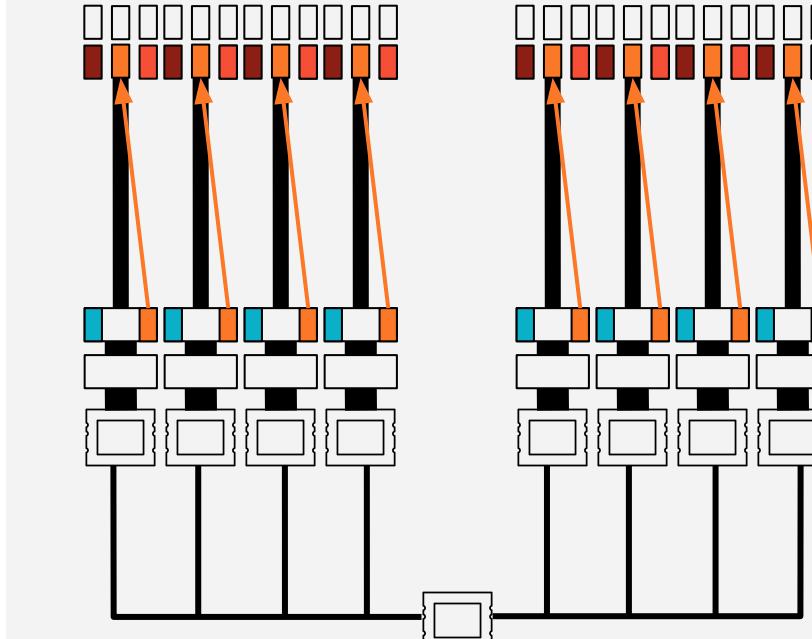
# Scaling in Time

Paging Costs



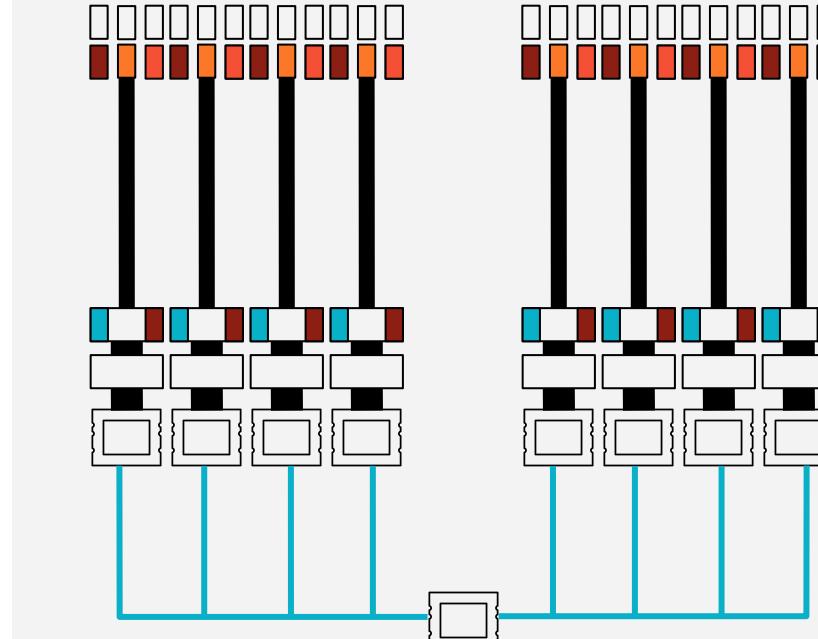
# Scaling in Time

Paging Costs



# Scaling in Time

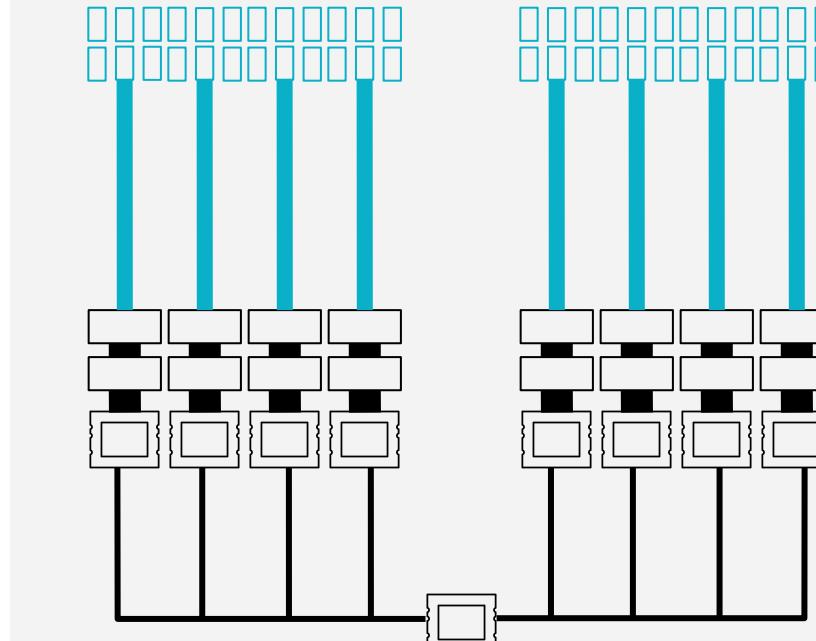
Paging Costs



# Scaling in Time

## Paging Costs

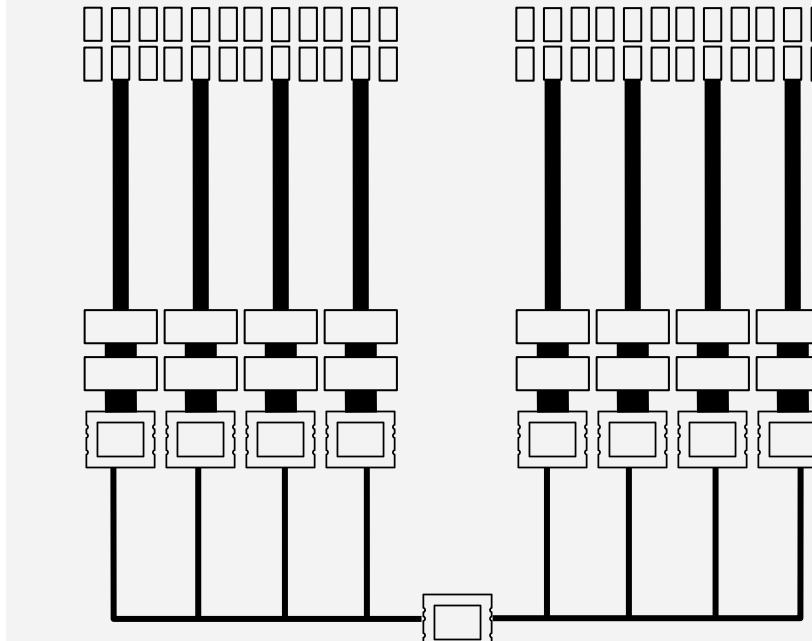
- Most cycles spent paging matrices in/out of HBM:
  - ✖ Low bandwidth  
(1/100X on-chip SRAM)
  - ✖ **High access latency**  
(300ns-1300ns)
  - ✖ **High access power**  
(4-6 pJ/bit for R/W)



# Scaling in Time

## Paging Costs

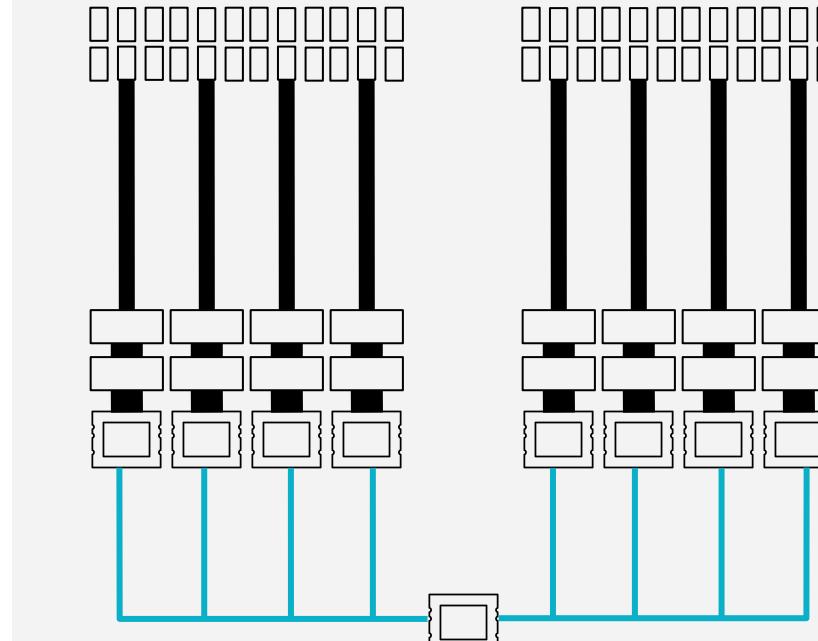
- Most cycles spent paging matrices in/out of HBM:
  - ✖ Low bandwidth  
(1/100X on-chip SRAM)
  - ✖ **High access latency**  
(300ns-1300ns)
  - ✖ **High access power**  
(4-6 pJ/bit for R/W)
- **High Batch** (100s-1000s)  
to saturate compute for  
GEMV at **output  
generation.**



# Scaling in Time

## Paging Costs

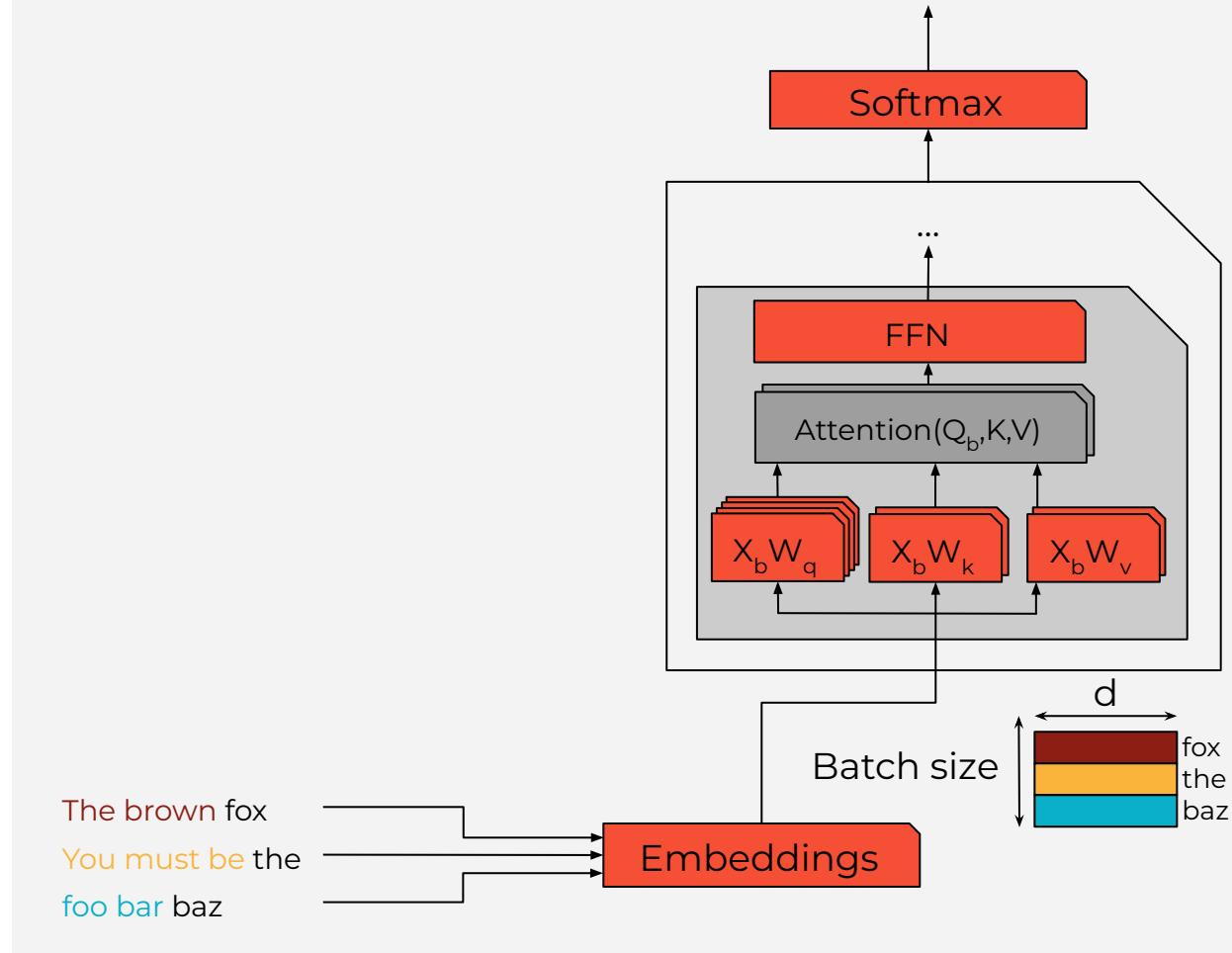
- Most cycles spent paging matrices in/out of HBM:
  - ✖ Low bandwidth  
(1/100X on-chip SRAM)
  - ✖ **High access latency**  
(300ns-1300ns)
  - ✖ **High access power**  
(4-6 pJ/bit for R/W)
- **High Batch** (100s-1000s)  
to saturate compute for  
GEMV at **output  
generation.**
- **Scaling Challenges**



# Batching

Increasing output gen. operational intensity

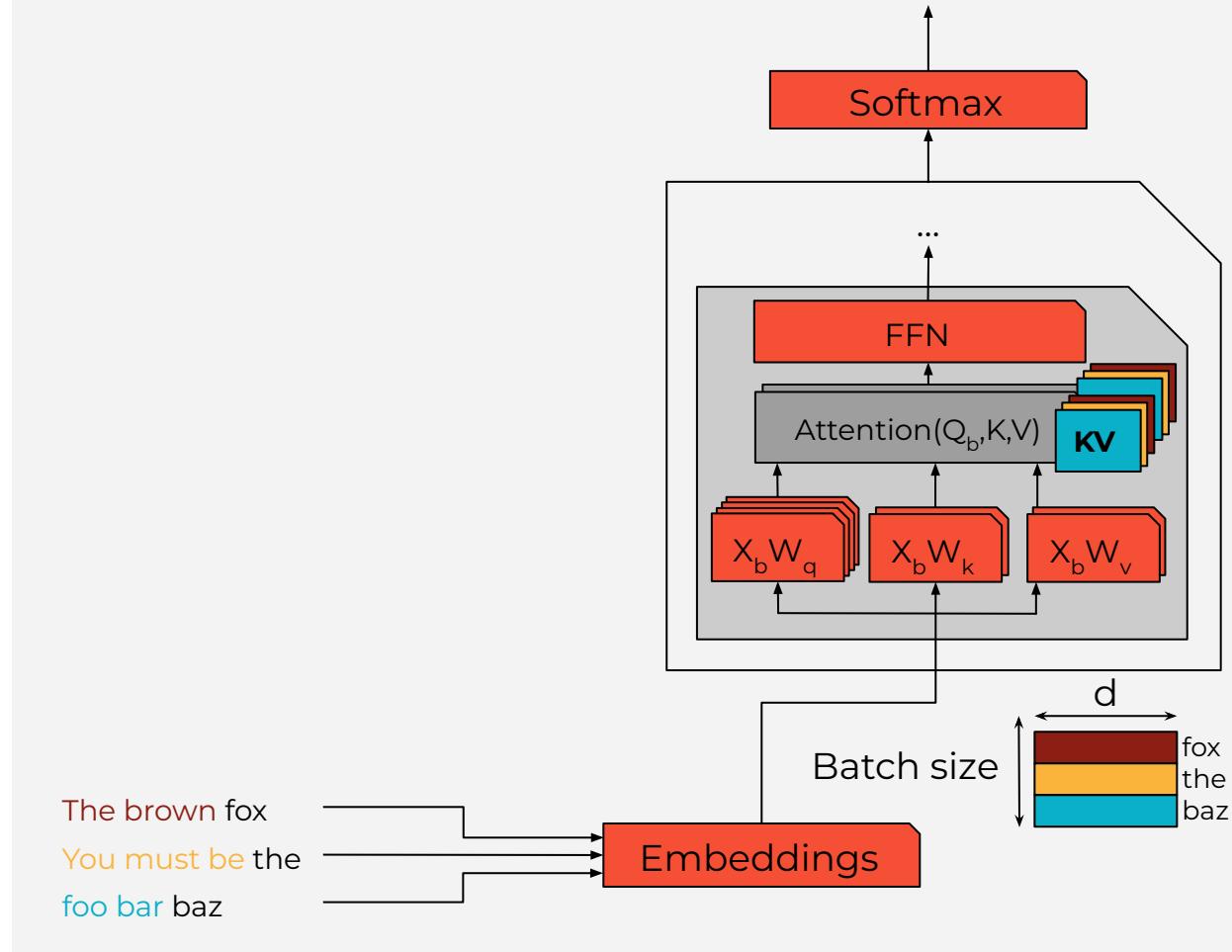
- Weights can be shared
  - ✓ GEMM



# Batching

Increasing output gen. operational intensity

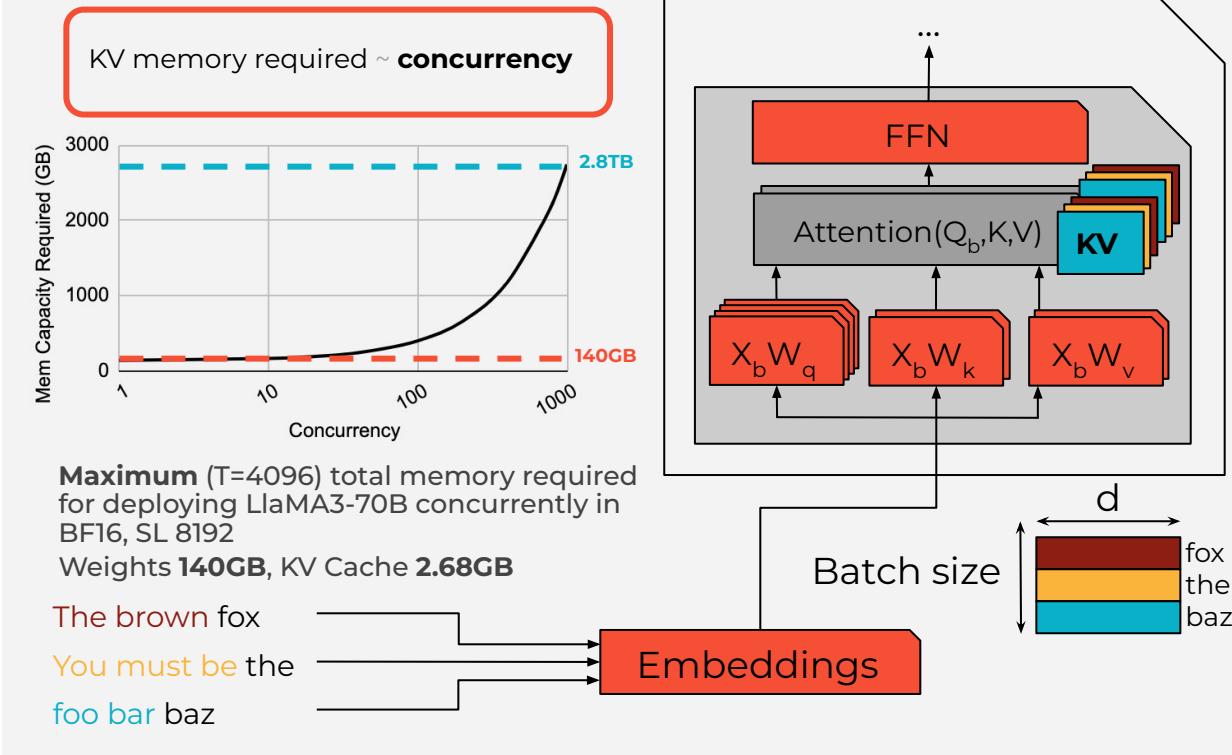
- Weights can be shared
  - ✓ GEMM
- KV Caches **can not be shared**
  - ✗ Irreducibly GEMV



# Batching

Increasing output gen. operational intensity

- Weights can be shared
  - ✓ GEMM
- KV Caches **can not be shared**
  - ✗ Irreducibly GEMV
- Batching ↑
  - (non-attention) Utilization ↑
  - Concurrency ↑
  - Memory capacity req. ↑  
(Latency ↑?)



# Batching

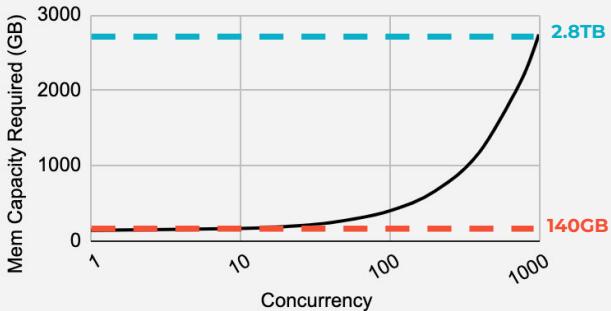
Increasing output gen. operational intensity

- Weights can be shared
  - ✓ GEMM
- KV Caches **can not be shared**
  - ✗ Irreducibly GEMV
- Batching ↑
  - (non-attention) Utilization ↑
  - Concurrency ↑
  - Memory capacity req. ↑  
(Latency ↑?)
- Low operational density  
(attention) share of the workload increases with CL

KV Cache size =

$$\text{numerics} * d_q * \mathbf{T} \quad \text{Matrix size}$$
$$* 2 \quad K \text{ and } V$$
$$* \mathbf{N} \quad \# \text{ decoder layers}$$
$$/ \mathbf{G} \quad \text{G-way GQA}$$

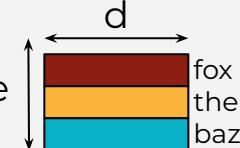
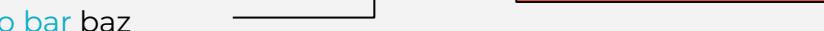
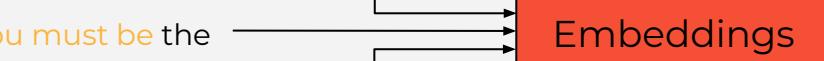
KV memory required ~ **concurrency**



**Maximum** ( $T=4096$ ) total memory required for deploying LLaMA3-70B concurrently in BF16, SL 8192  
Weights **140GB**, KV Cache **2.68GB**

The brown fox

You must be the  
foo bar baz

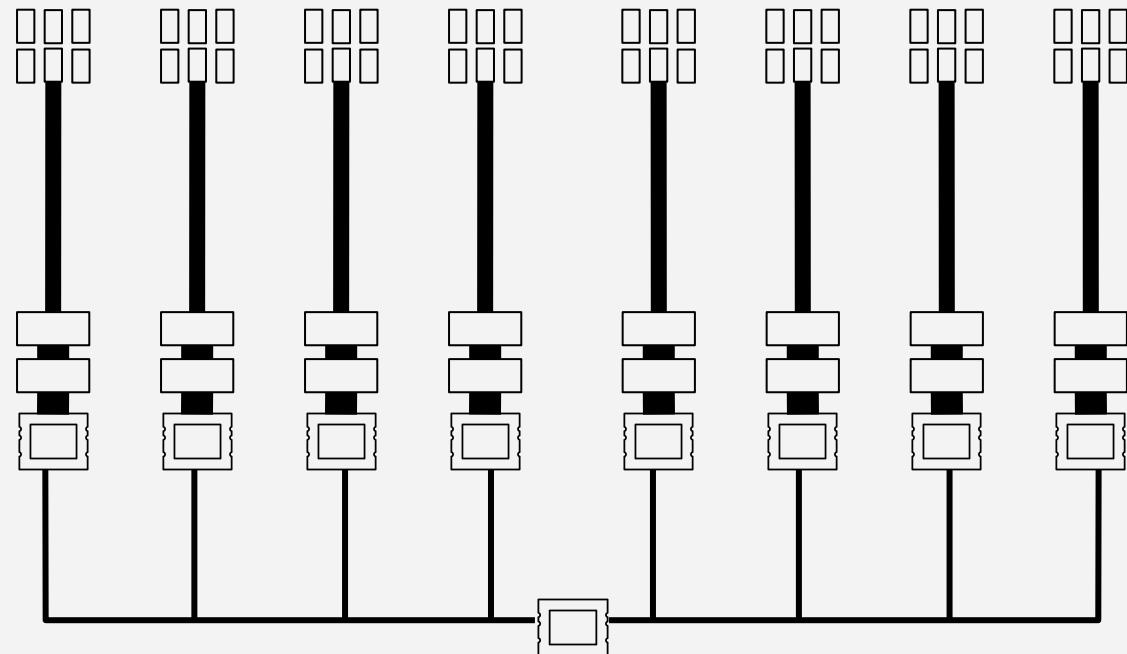


# Sharding (1)

## Scaling Challenges

Methods for increasing memory bandwidth:

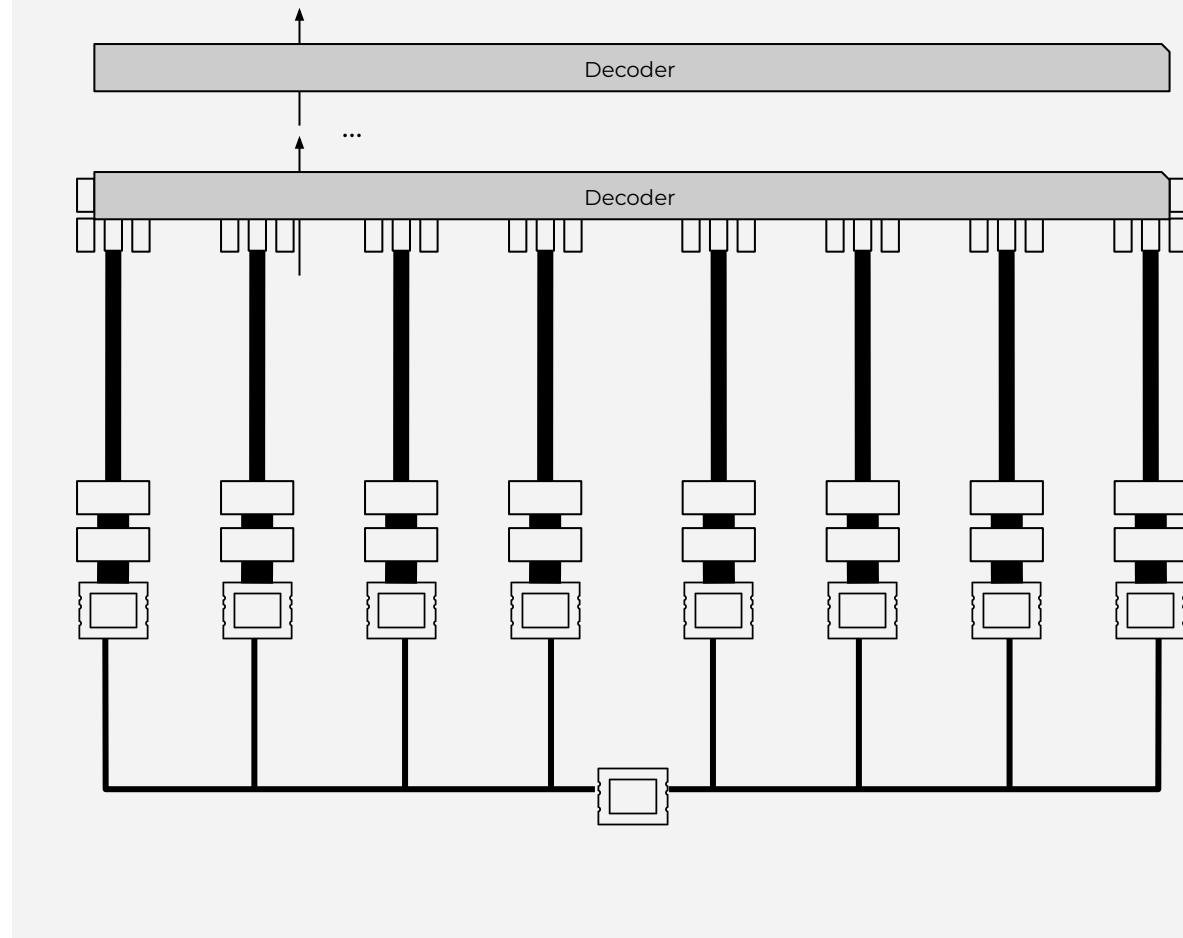
- Tensor Parallelism
- Pipeline Parallelism
- Expert Parallelism (MoE)
- Model Parallelism
- Data Parallelism (Training)



# Sharding (2)

Tensor Parallelism

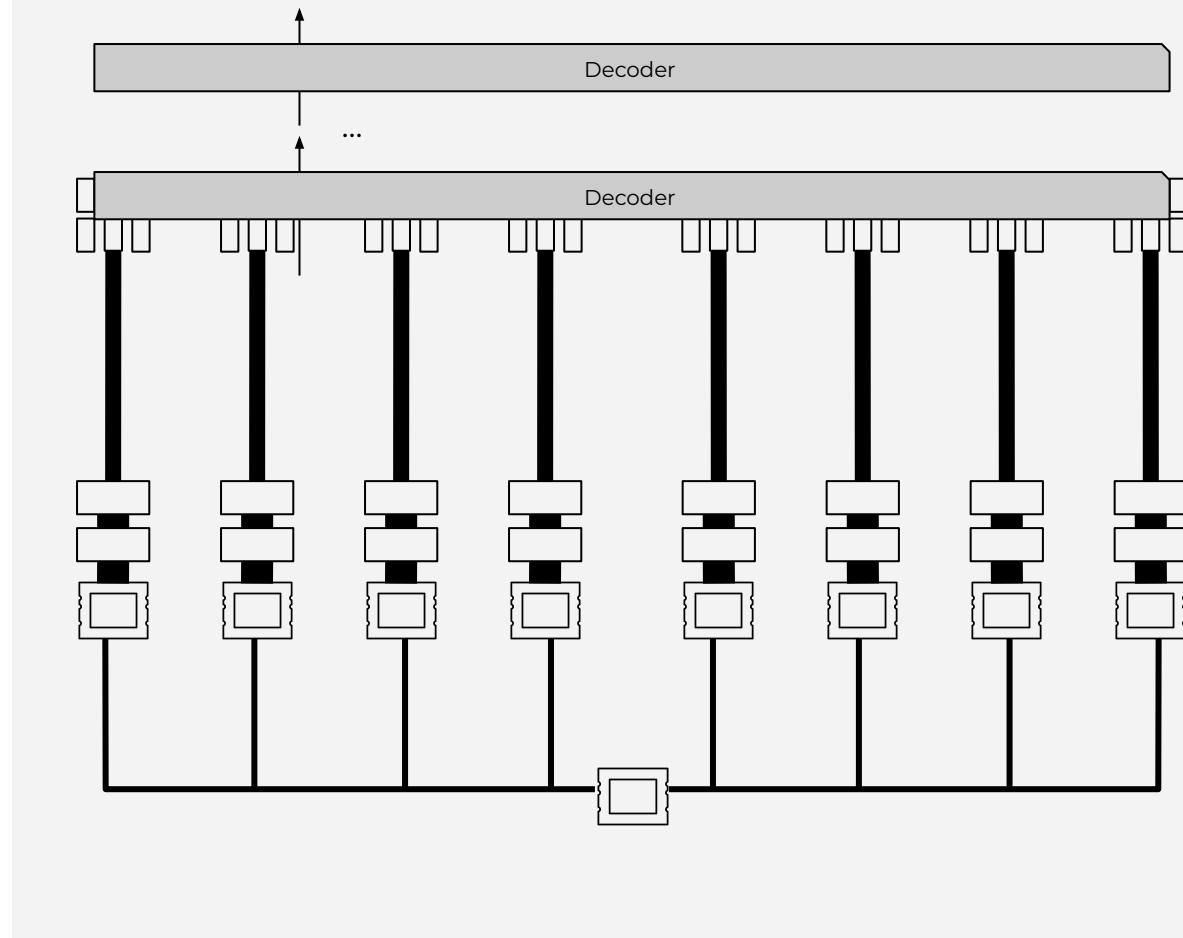
- Total mem. bw/capacity ↑  
Token rates ↑



# Sharding (2)

## Tensor Parallelism

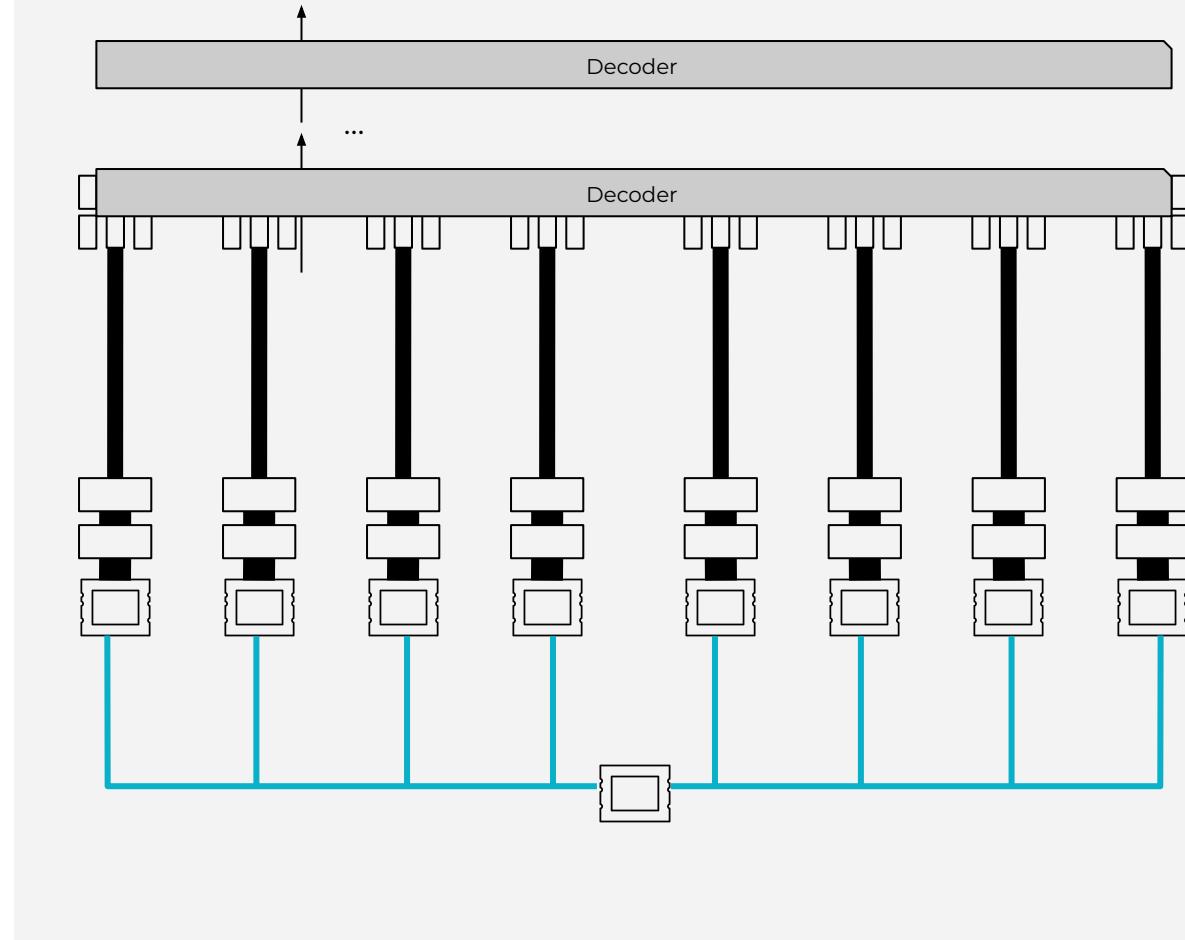
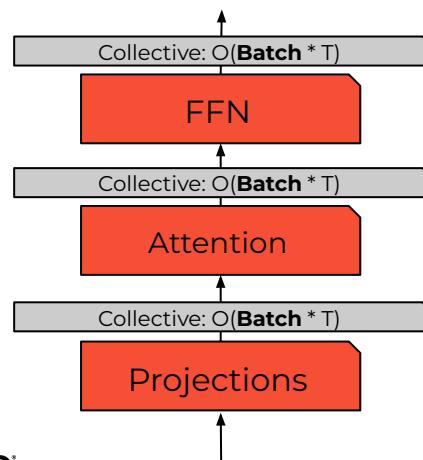
- Total mem. bw/capacity ↑
- Token rates ↑
- No utilization gains



# Sharding (2)

## Tensor Parallelism

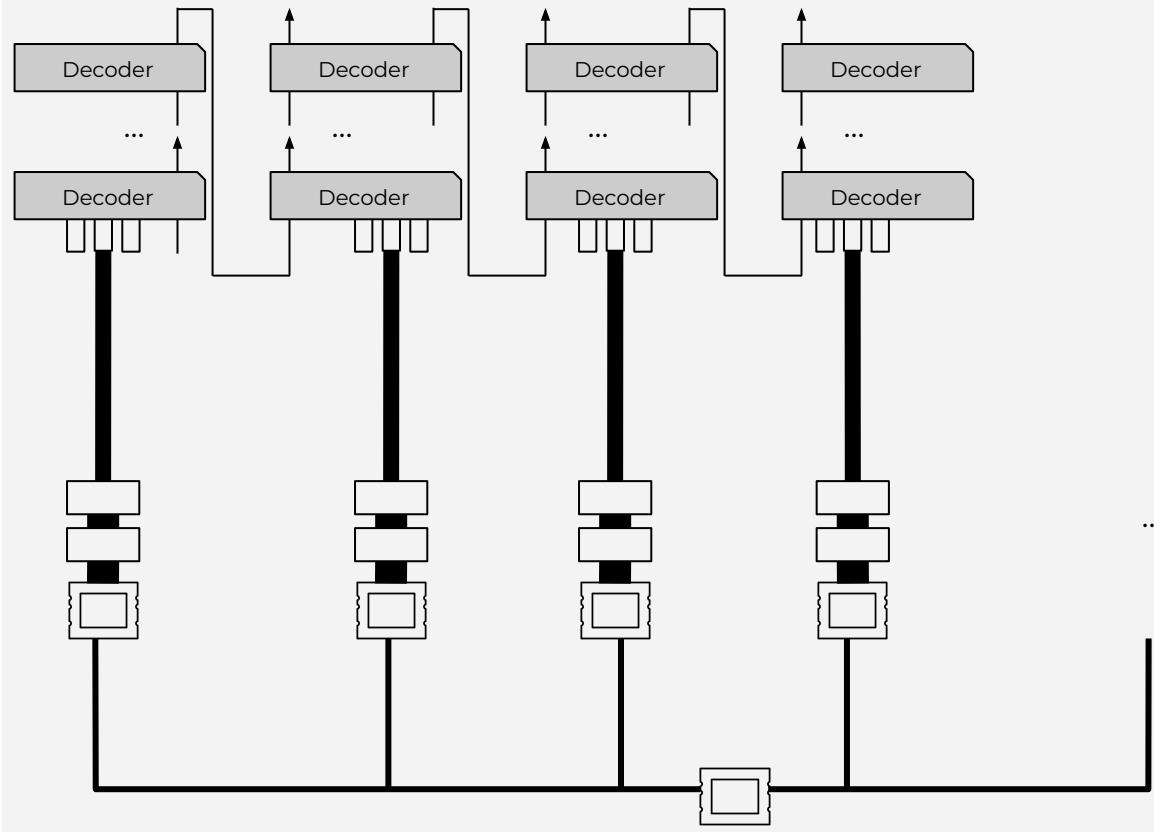
- Total mem. bw/capacity ↑
- Token rates ↑
- No utilization gains
- Expensive networking collectives limit width



# Sharding (3)

Pipeline Parallelism

- Total mem. bw/capacity ↑



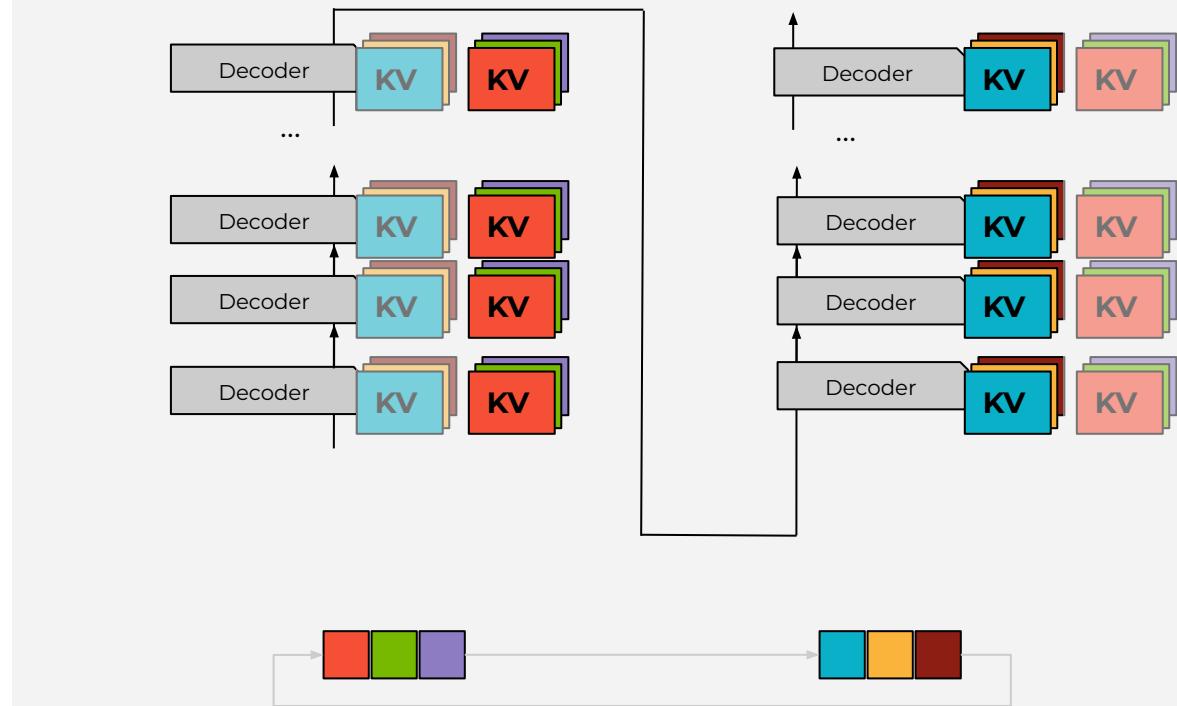
$$6 = 3 * 2$$

concurrency = **batch** \* pipelining  
 kv cache required = concurrency \* kv cache size

# Sharding (3)

Pipeline Parallelism

- Total mem. bw/capacity ↑
- Diminishing returns at output generation



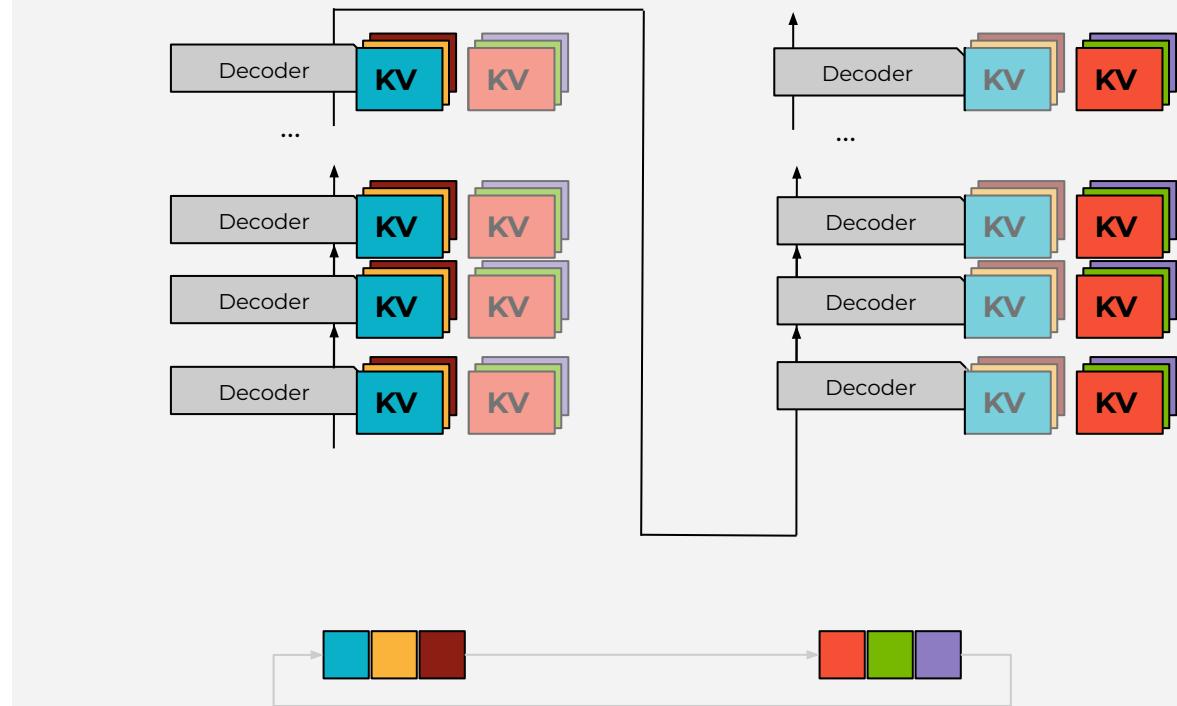
$$6 = 3 * 2$$

concurrency = **batch** \* pipelining  
 kv cache required = concurrency \* kv cache size

# Sharding (3)

Pipeline Parallelism

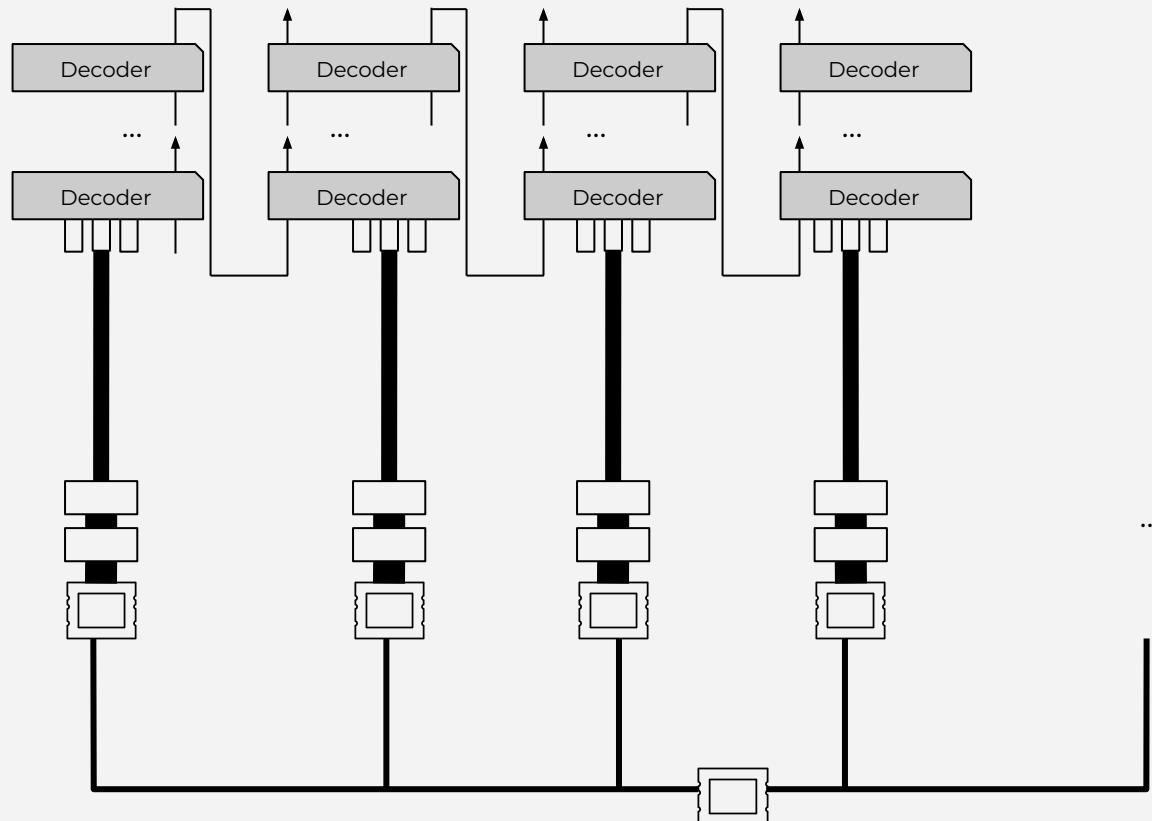
- Total mem. bw/capacity ↑
- Diminishing returns at output generation



# Sharding (3)

## Pipeline Parallelism

- Total mem. bw/capacity ↑
- Diminishing returns at output generation



concurrency = **batch** \* pipelining

kv cache required = concurrency \* kv cache size

# Sharding (3)

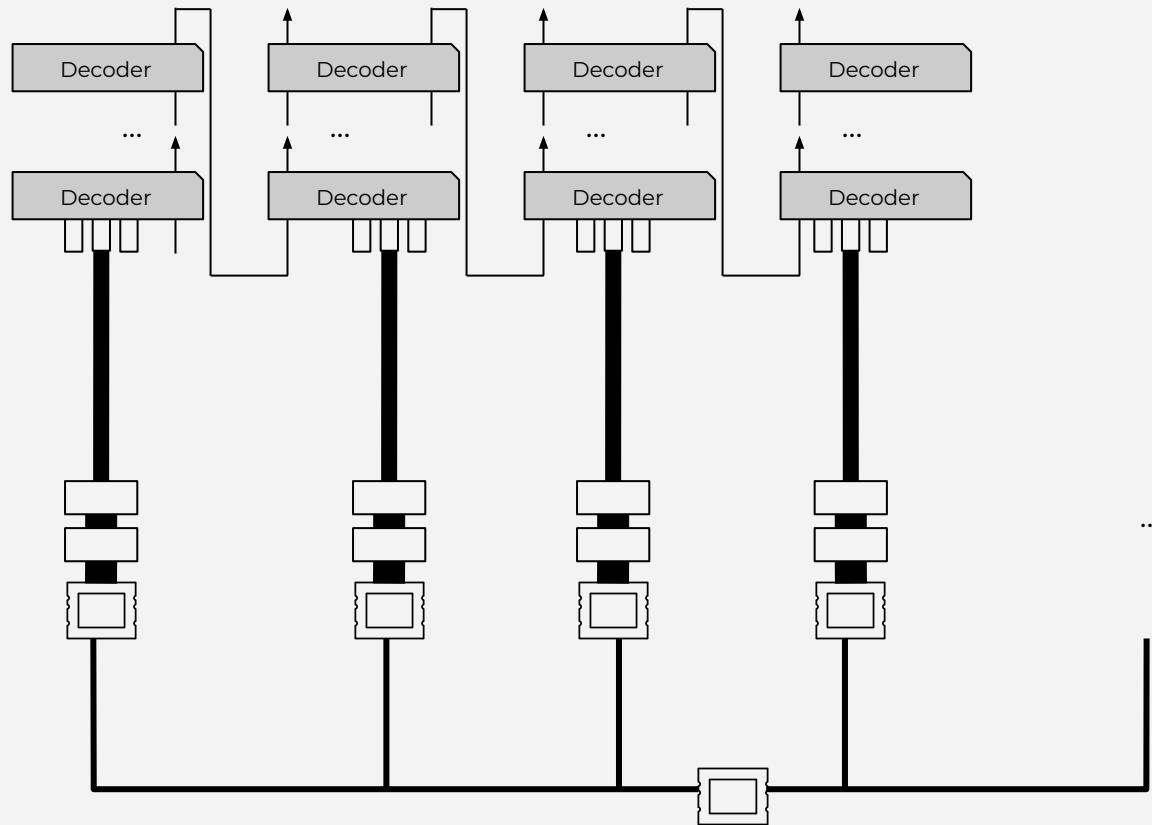
## Pipeline Parallelism

- Total mem. bw/capacity ↑
- Diminishing returns at output generation
- Still harmed by batch but lighter comms than TP

Total cost = # hops \* # tokens \* decoder transfer

Batch	1	100	1000
Total Cost (GB)	0.8	80	800

LLaMA3-70B decoder transfer costs for 10 Hops, 5000 output tokens

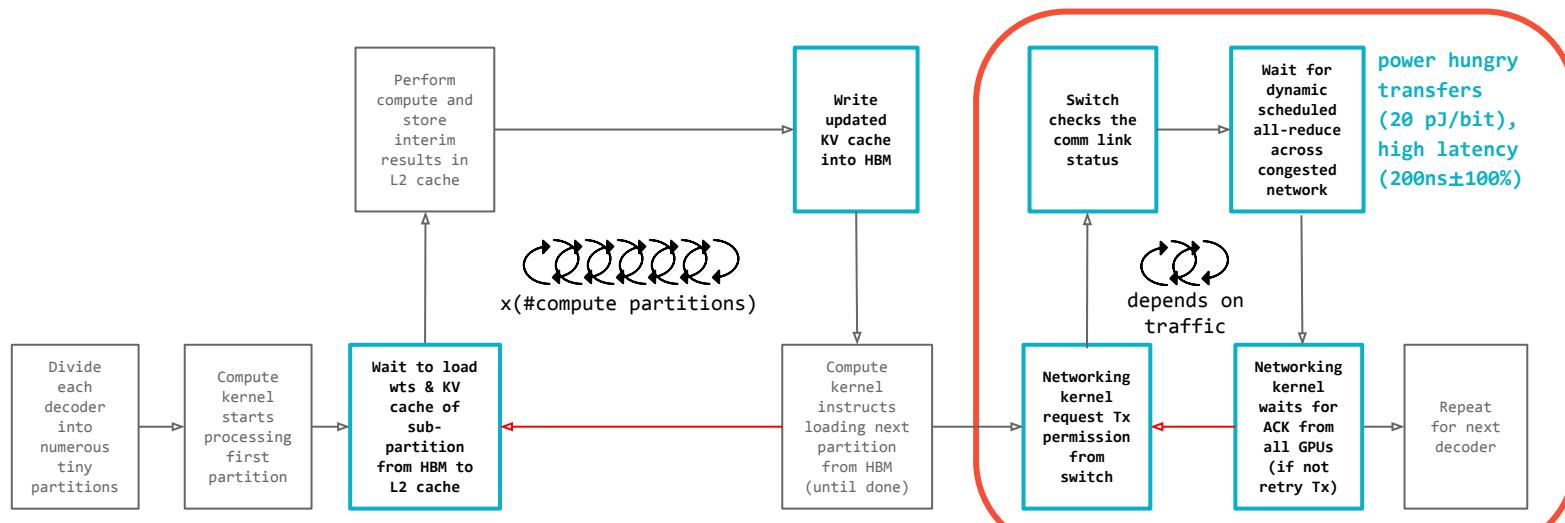


decoder transfer = numerics \* d \* **batch**

concurrency = **batch** \* pipelining

kv cache required = concurrency \* kv cache size

# GPU LLM Inference Overview

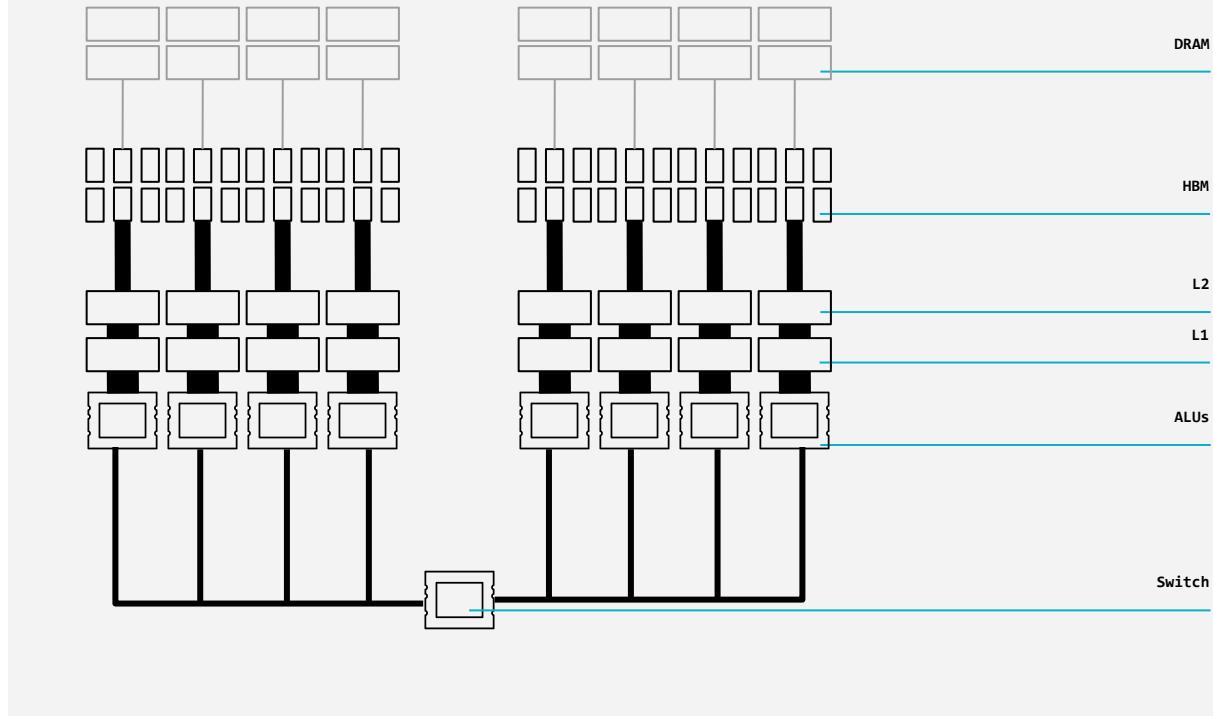


Poor off-chip HBM memory BW (high latency  
~300ns-1300ns), very high cost (HBM<sup>s</sup>/interposers),  
very power hungry (4-6 pJ/bit for read/write)

# Summary

Scaling High-Batch (mostly) in Time

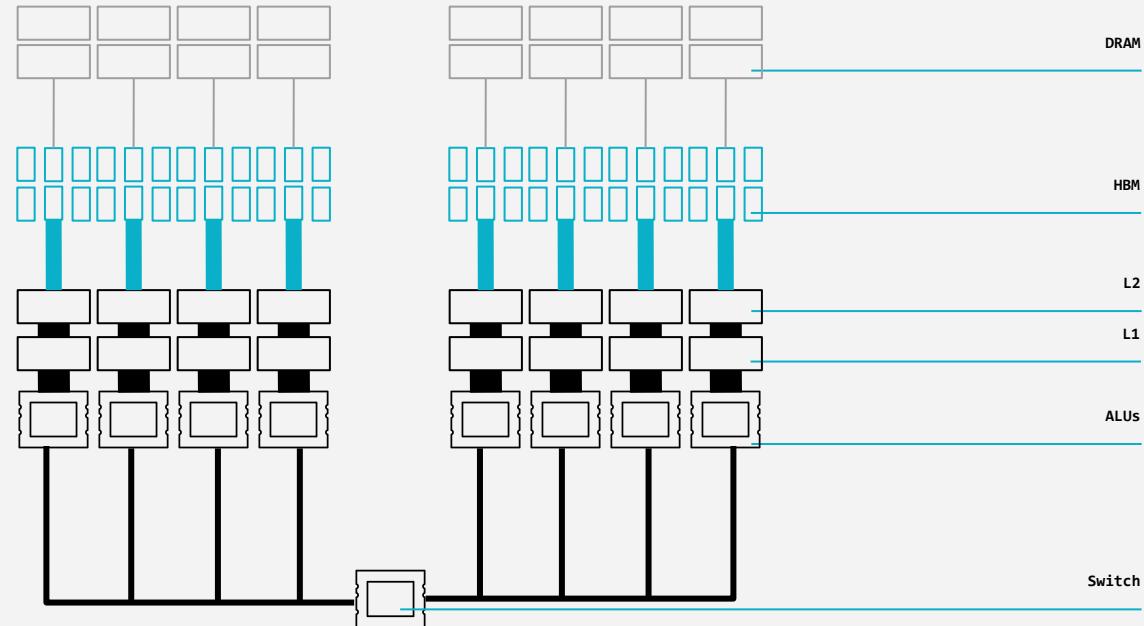
- **Token rates and model size** limitations



# Summary

Scaling High-Batch (mostly) in Time

- **Token rates** and **model size** limitations
- **High-Batch** mandatory



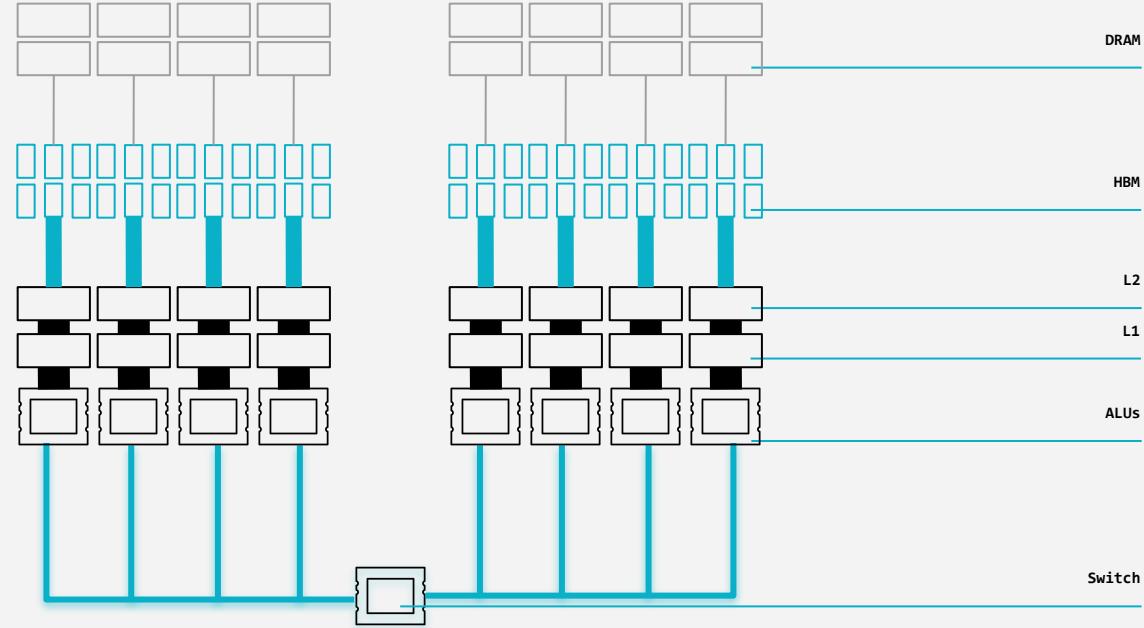
Non-linear Throughput Scaling:

<https://www.run.ai/blog/achieve-2x-inference-throughput-reduce-latency-leveraging-multi-gpu>

# Summary

Scaling High-Batch (mostly) in Time

- **Token rates** and **model size** limitations
- **High-Batch** mandatory
- **Scaling** challenge



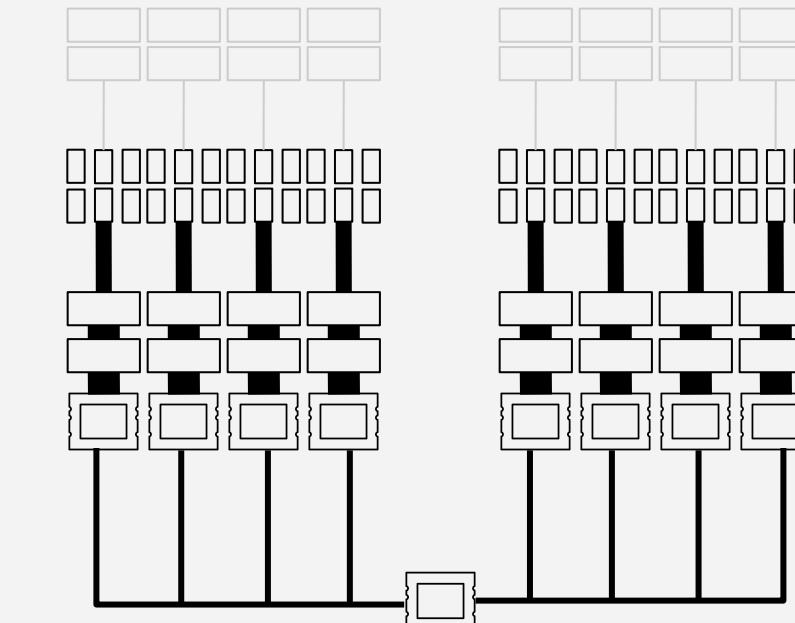
Non-linear Throughput Scaling:

<https://www.run.ai/blog/achieve-2x-inference-throughput-reduce-latency-leveraging-multi-gpu>

# Tabula Rasa

Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

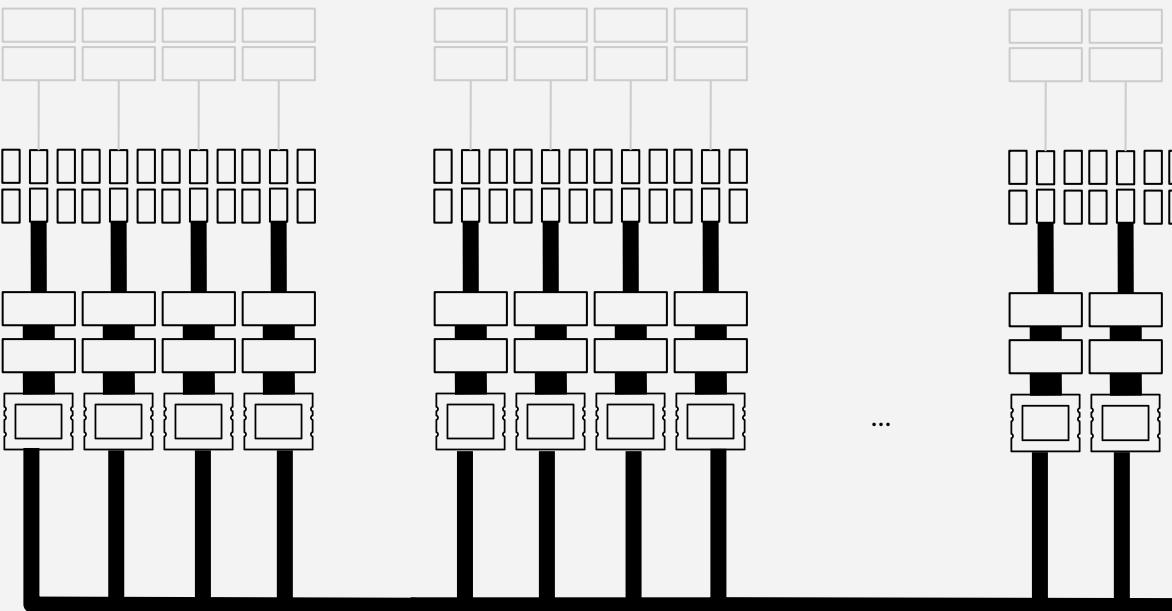


# Tabula Rasa

Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips

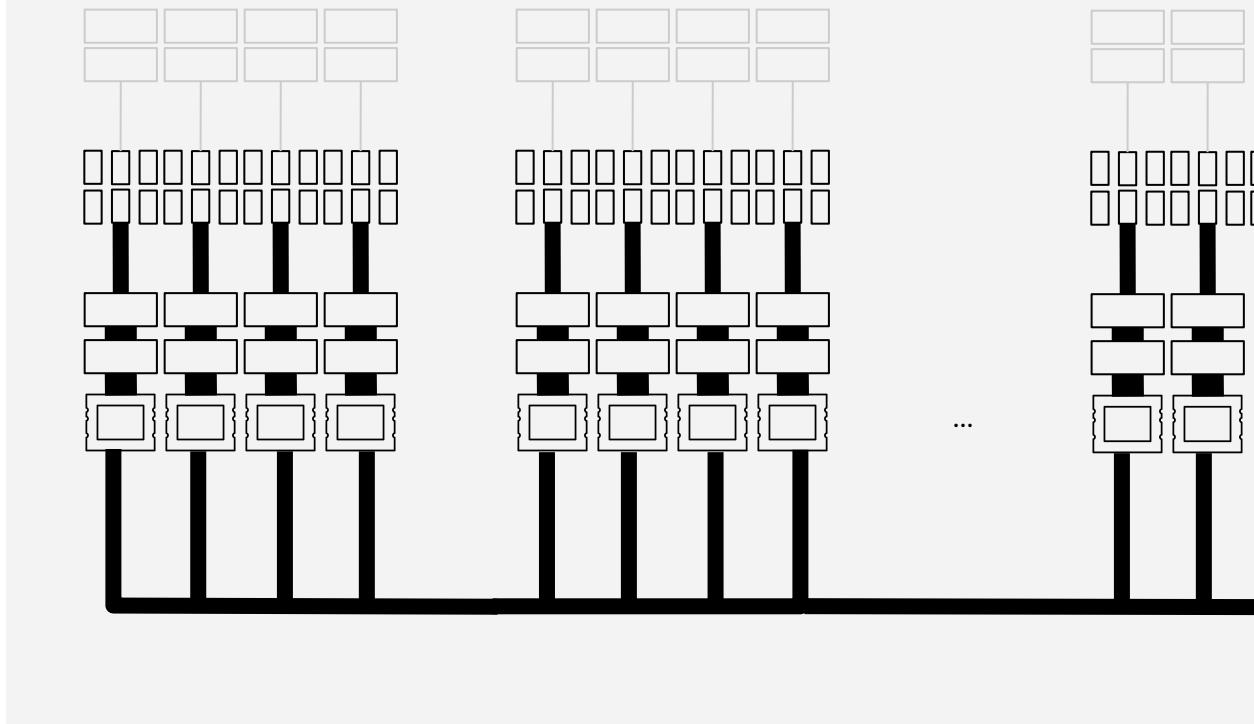


# Tabula Rasa

Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑



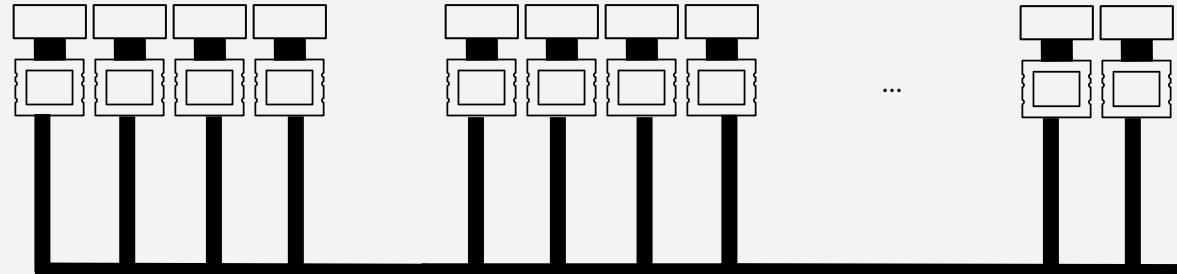
# Tabula Rasa

Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑

Enough SRAM capacity, **drop HBM?**



# Tabula Rasa

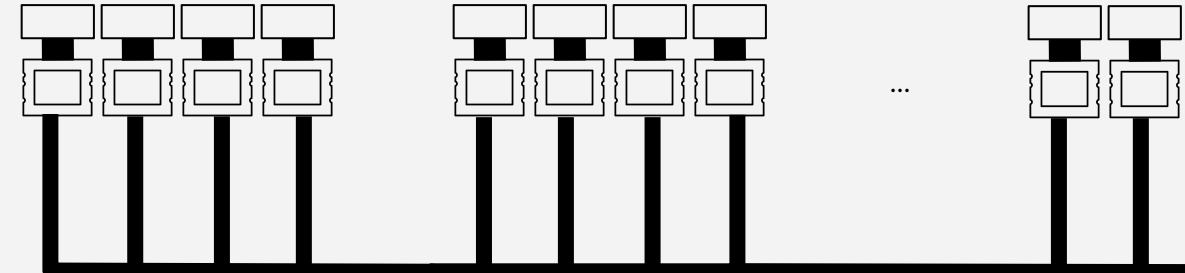
Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑

Enough SRAM capacity, **drop HBM?**

- ✓ **GEMV+GEMM Utilization ↑**



# Tabula Rasa

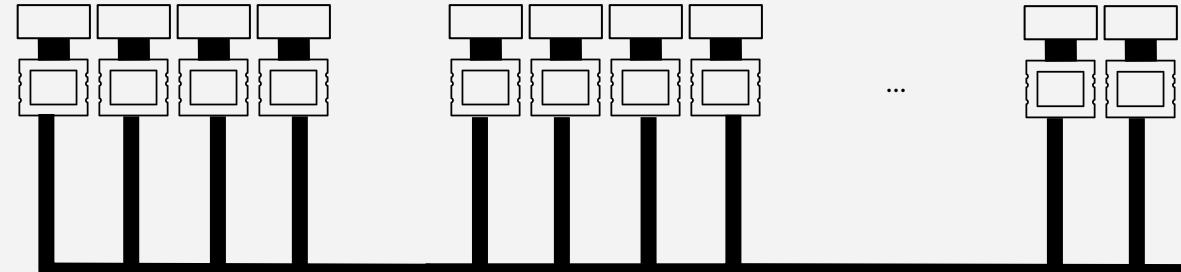
Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑

Enough SRAM capacity, **drop HBM?**

- ✓ **GEMV+GEMM Utilization** ↑
- ✓ Batch/KV Memory requirements ↓



# Tabula Rasa

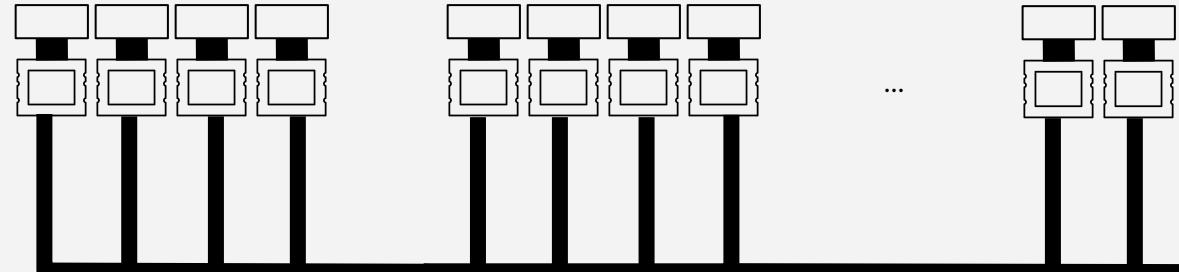
Making KV Caches worthy of the name

Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑

Enough SRAM capacity, **drop HBM?**

- ✓ **GEMV+GEMM Utilization** ↑
- ✓ Batch/KV Memory requirements ↓
- ✓ Output Token Rates ↑



# Tabula Rasa

Making KV Caches worthy of the name

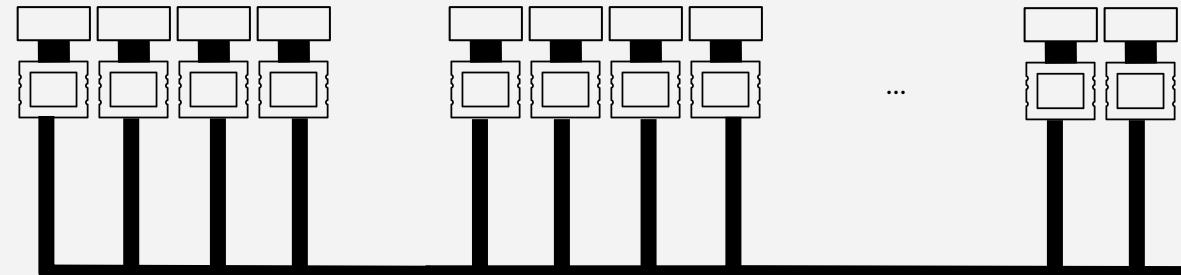
Networking advantages in  
**latency, bandwidth, scale?**

- ✓ Hop/collective costs ↓
- ✓ Sharding on more chips
- ✓ Aggressive Tensor Parallelism
- ✓ Input, Output Token Rates ↑

Enough SRAM capacity, **drop HBM?**

- ✓ **GEMV+GEMM Utilization** ↑
- ✓ Batch/KV Memory requirements ↓
- ✓ Output Token Rates ↑

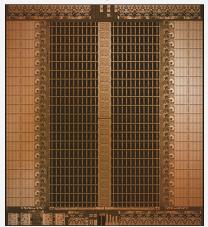
Best of both worlds in terms of  
**utilization** and **token rates**





# Language Processing Units (LPUs)

↑ Applying our Solution Sketch



## GroqChip™

The purpose-built  
Language Processing  
Unit™ accelerator



## GroqCard™

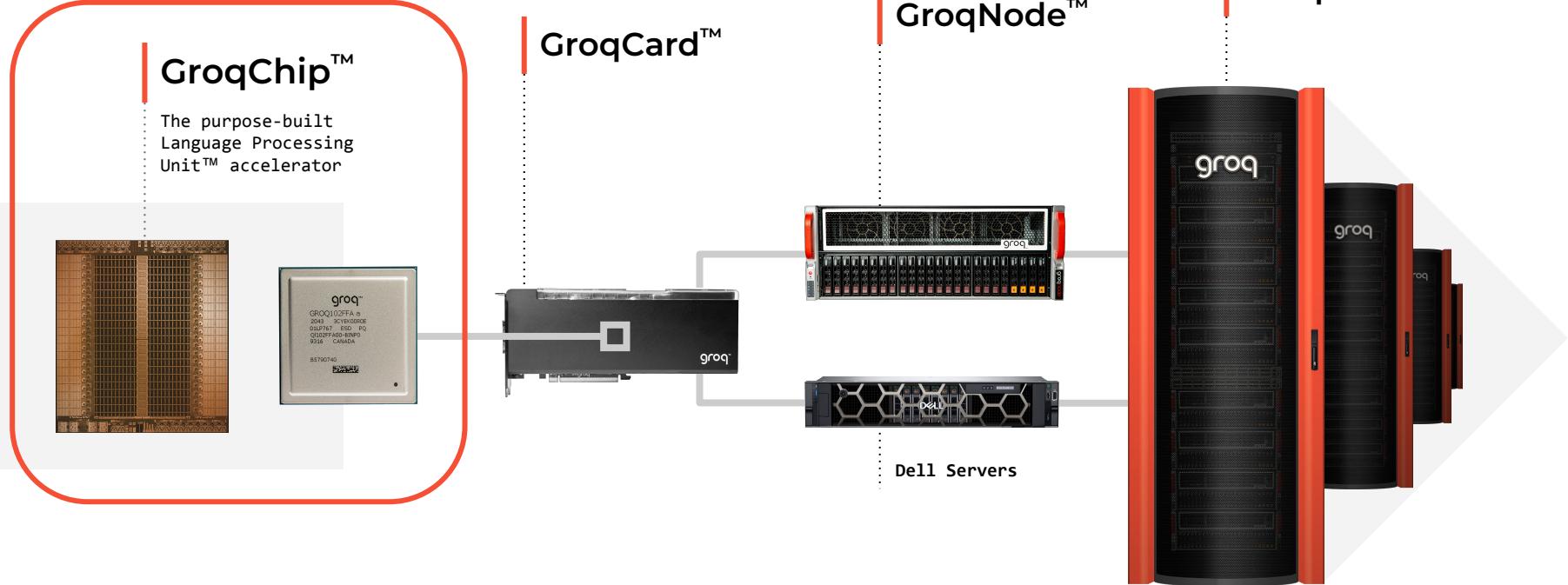
GroqNode™



Dell Servers

## GroqRack™

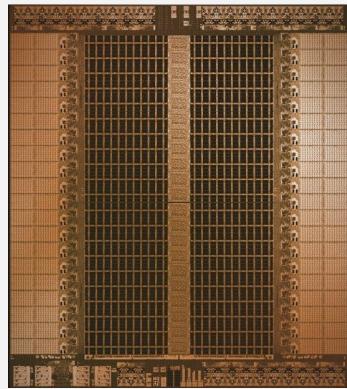




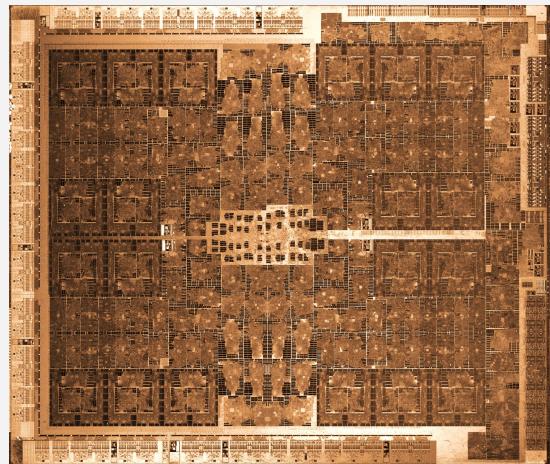
# LPU Overview

SRAM, predictability, programmability

## Language Processing Unit™



## Typical Graphics Processing Unit



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

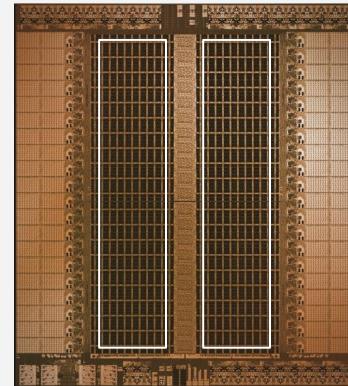
# LPU Overview

SRAM, predictability, programmability

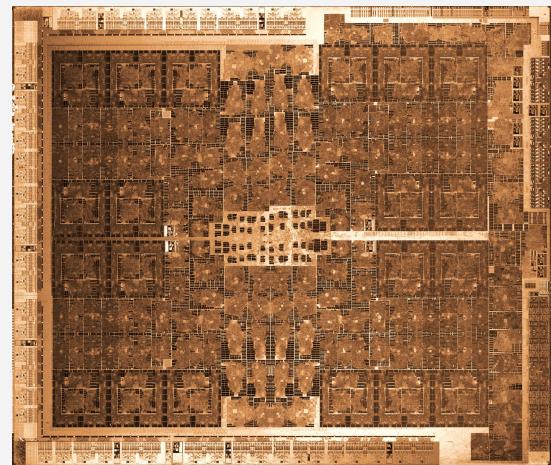
## ■ SRAM:

- ✓ No hierarchy
- ✓ High concurrency

## Language Processing Unit™



## Typical Graphics Processing Unit



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Overview

SRAM, predictability, programmability

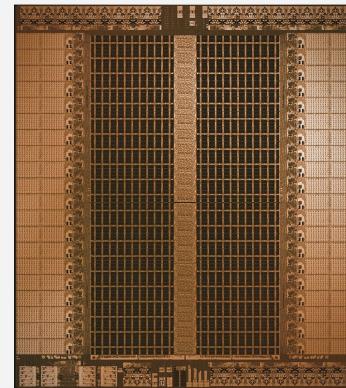
- **SRAM:**

- ✓ No hierarchy
- ✓ High concurrency

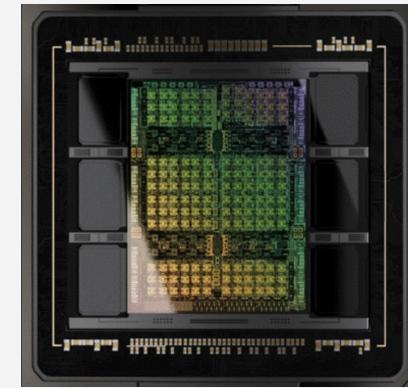
- **Deterministic:**

- ✓ Behavior known to “ns”
- ✓ Zero tail latency

## Language Processing Unit™



## Typical Graphics Processing Unit



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Overview

SRAM, predictability, programmability

- **SRAM:**
  - ✓ No hierarchy
  - ✓ High concurrency
- **Deterministic:**
  - ✓ Behavior known to “ns”
  - ✓ Zero tail latency
- Programmable at **high granularity**:
  - ✓ Latency advantage

## Language Processing Unit™



## Typical Graphics Processing Unit



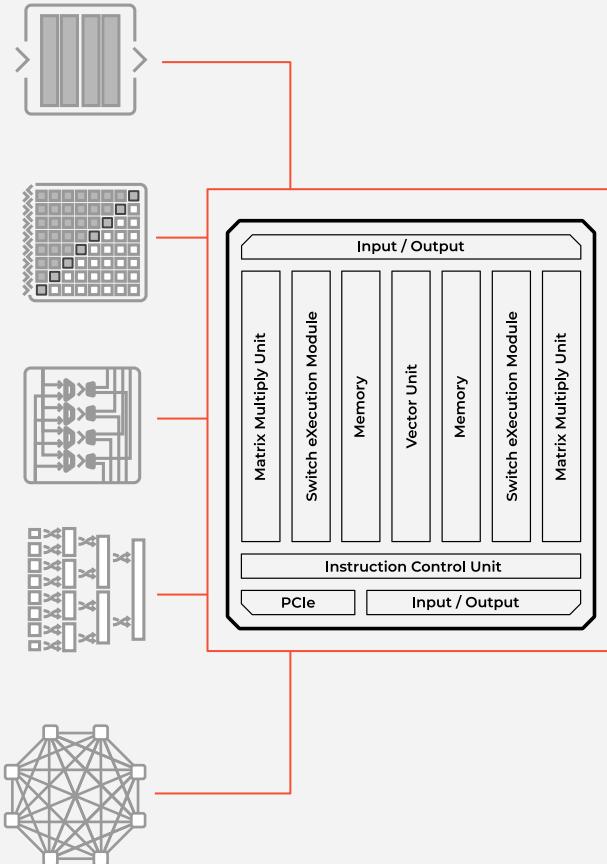
[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# V1 Specs

Specs for our 14nm GroqChip

- 230MB, massive memory concurrency
- Matrix and vector processing focused
- C2C networking

SRAM  
230MB capacity  
**80 TB/s BW**



Groq TruePoint™ Matrix  
750 TOP/s int8, 188 TFLOP/s fp16

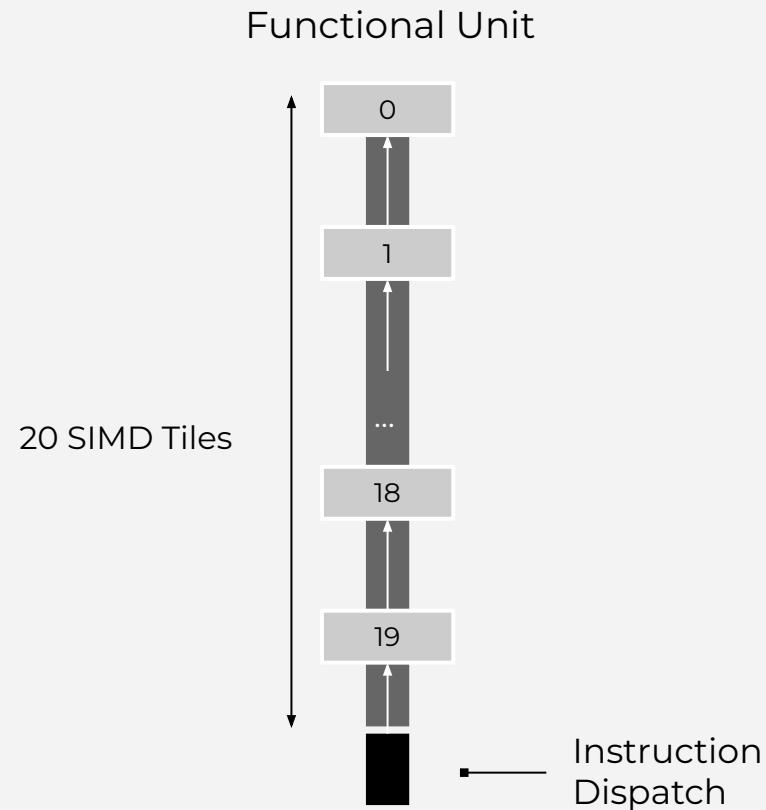
Programmable Vector Units  
5,120 Vector ALUs

Data Switch  
Shift, Transpose, Permuter

Statically Scheduled Network  
480 GB/s chip-to-chip (C2C) bandwidth

# LPU Layout

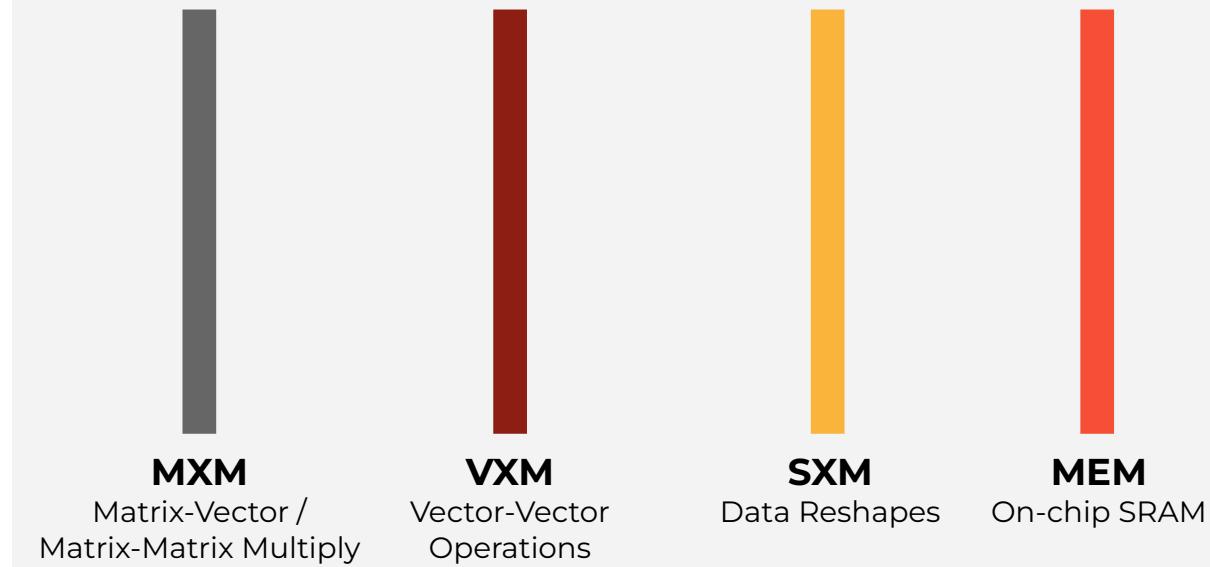
- SIMD building block



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

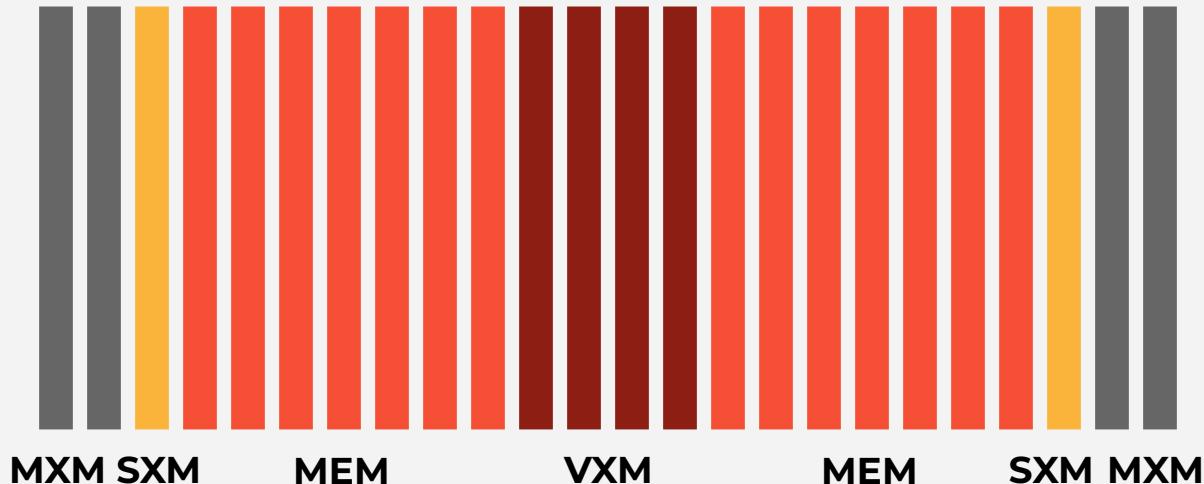
- SIMD building block
- Specialized to Functional Unit (FU)



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

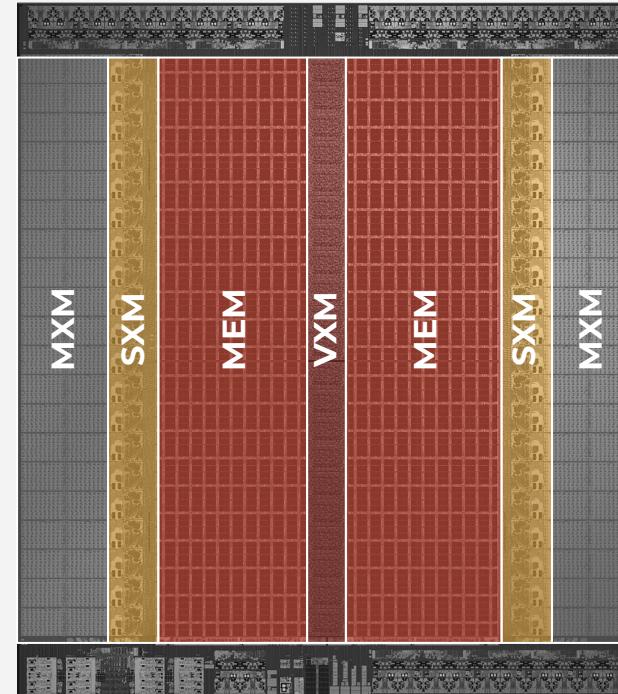
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

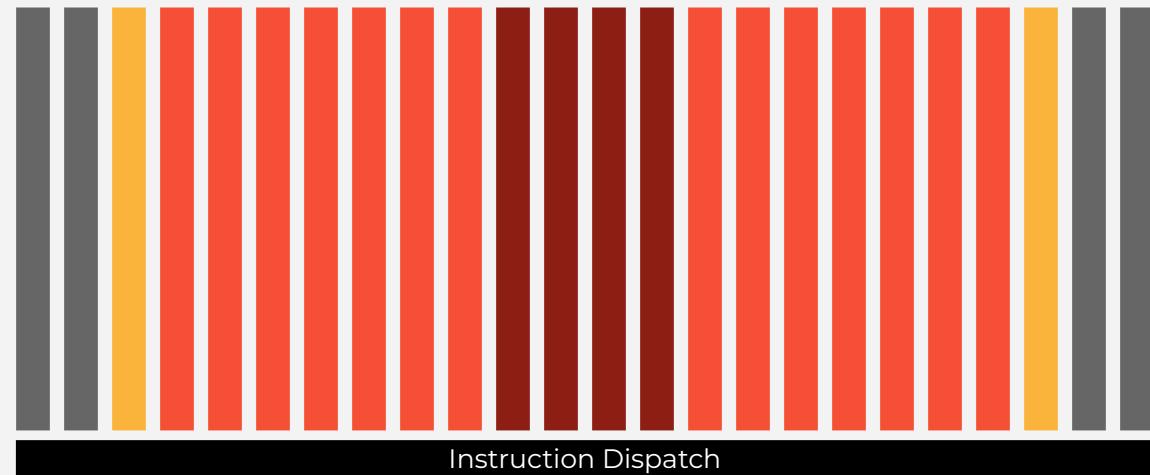
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

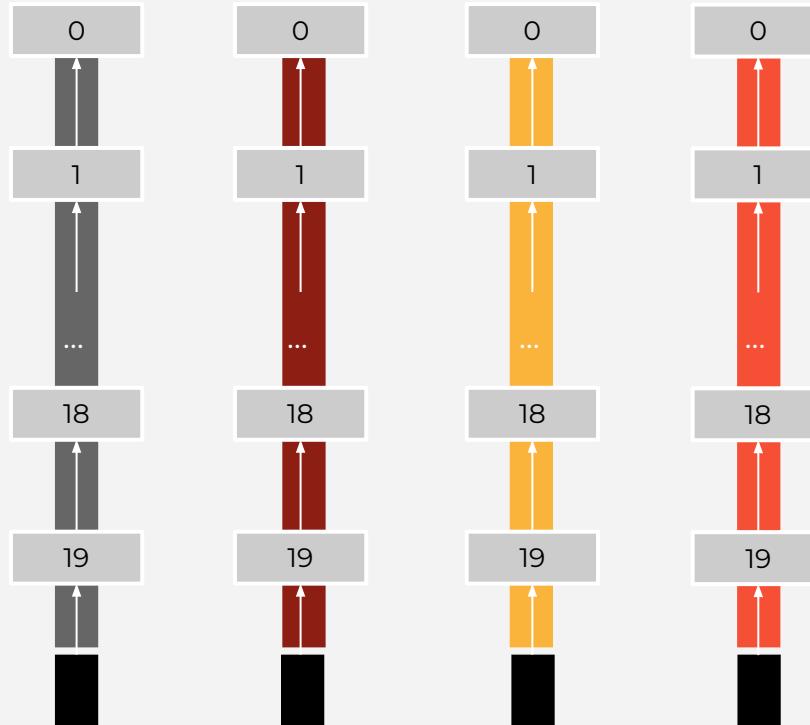
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

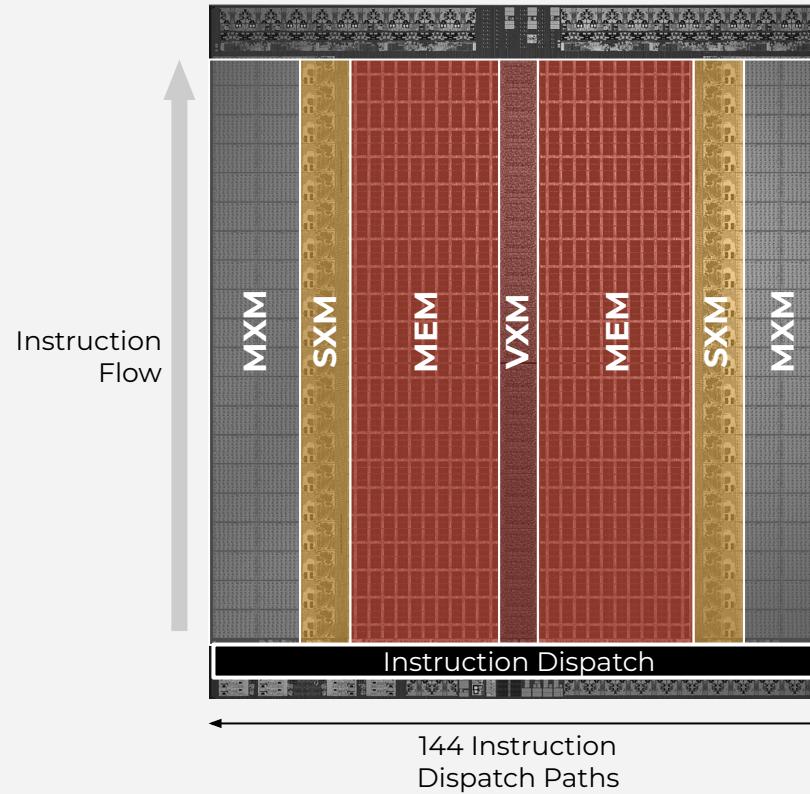
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

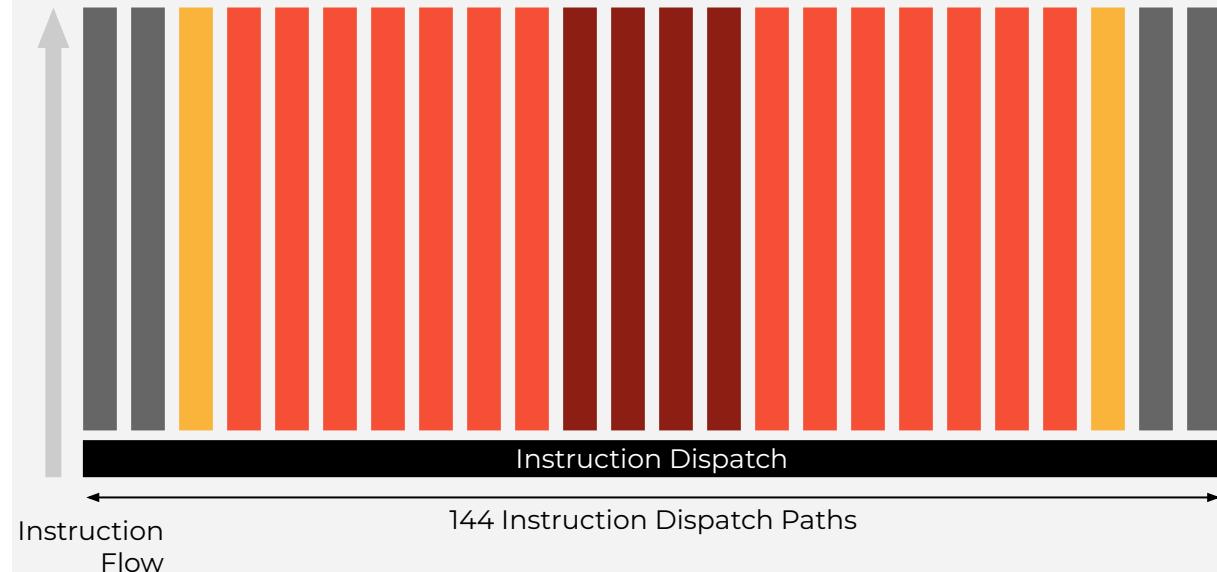
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

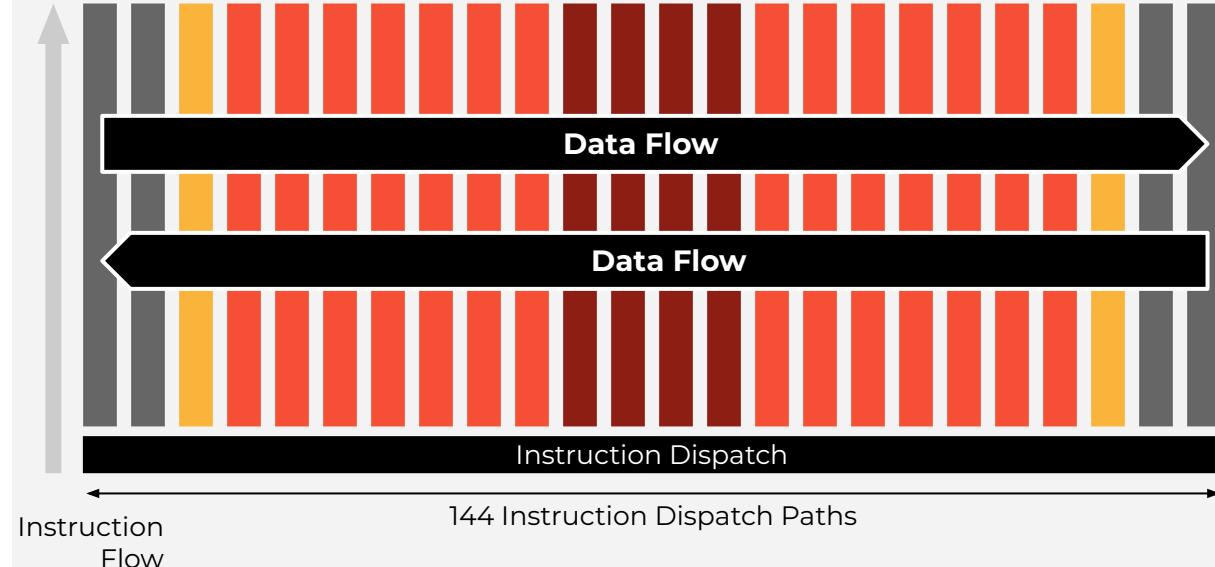
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

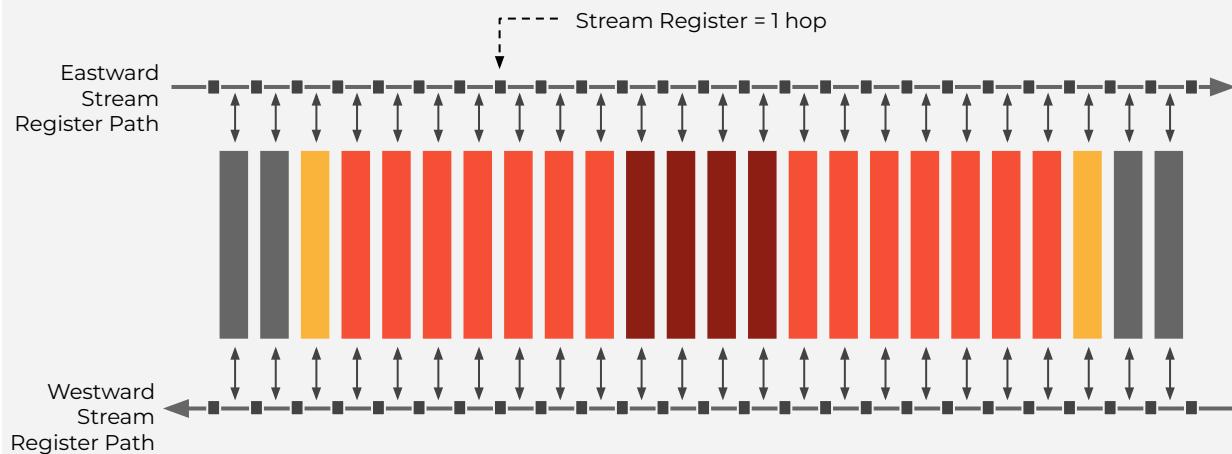
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**
- Connected via **stream register**-based 1D interconnect



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**
- Connected via **stream register**-based 1D interconnect



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

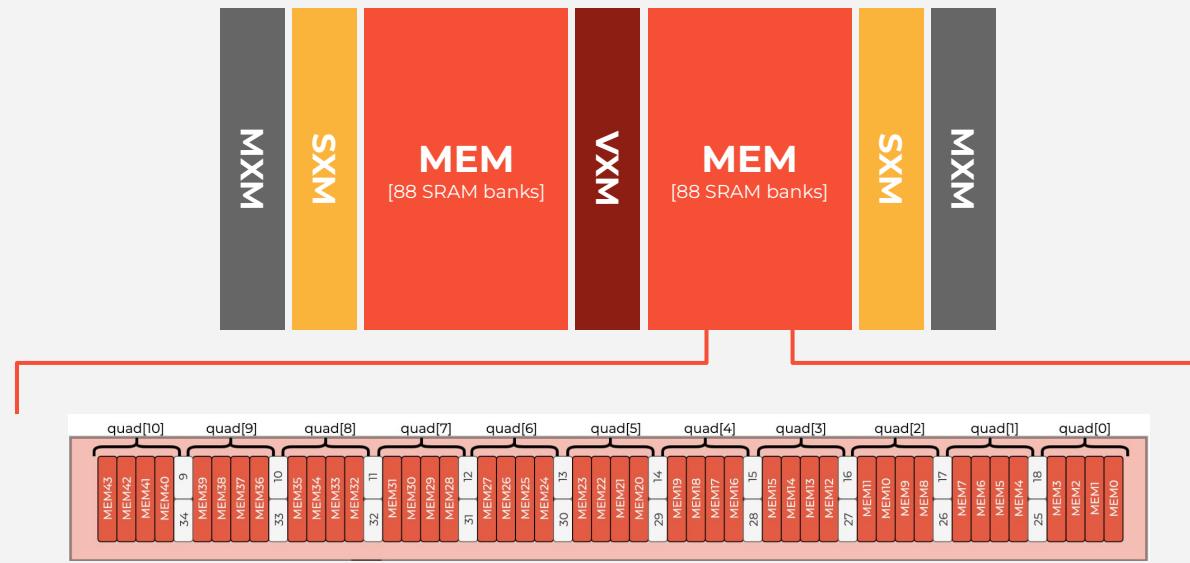
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**
- Connected via **stream register**-based 1D interconnect

Function	Instruction
ICU	NOP $N$ Ifetch Sync Notify Config Repeat $n, d$
MEM	Read $a, s$ Write $a, s$ Gather $s, map$ Scatter $s, map$ Countdown $d$ Step $a$ Iterations $n$
VXM	unary operation binary operation type conversions ReLU Tanh Exp RSqrt
MXM	LW IW ABC ACC
SXM	Shift up/down $N$ Permute $map$ Distribute $map$ Rotate $stream$ Transpose sg16
C2C	Deskew Send Receive

[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

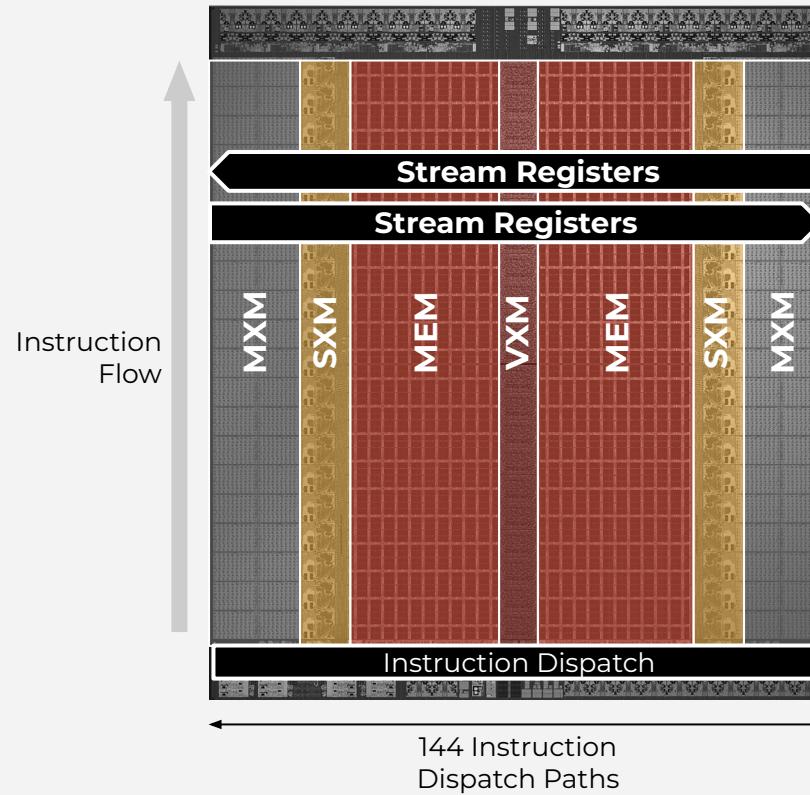
- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**
- Connected via **stream register**-based 1D interconnect



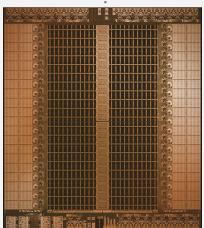
[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.

# LPU Layout

- SIMD building block
- Specialized to Functional Unit (FU)
- Stamped out for **concurrency** and **pipelining**
- Independently programmed in **lockstep**
- Connected via **stream register**-based 1D interconnect



[5]: Abts, Dennis, et al. "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads." *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2020.



## GroqChip™

The purpose-built  
Language Processing  
Unit™ accelerator



## GroqCard™

## GroqNode™

## GroqRack™



Dell Servers



# Networking

Synchronous and Statically Scheduled

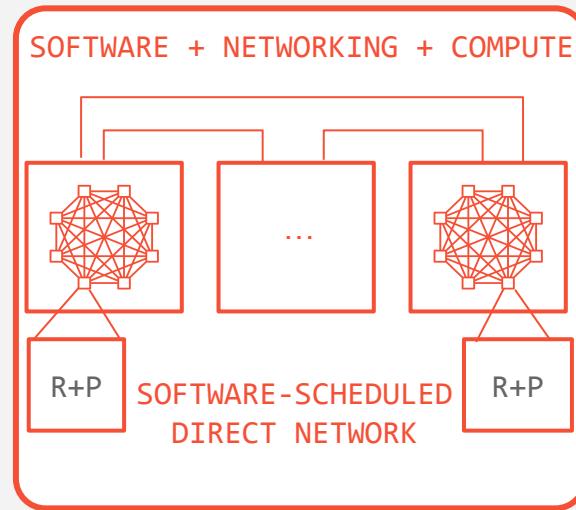
Recipe:

- n LPUs
- ~16n cables
- 1 amazing software stack

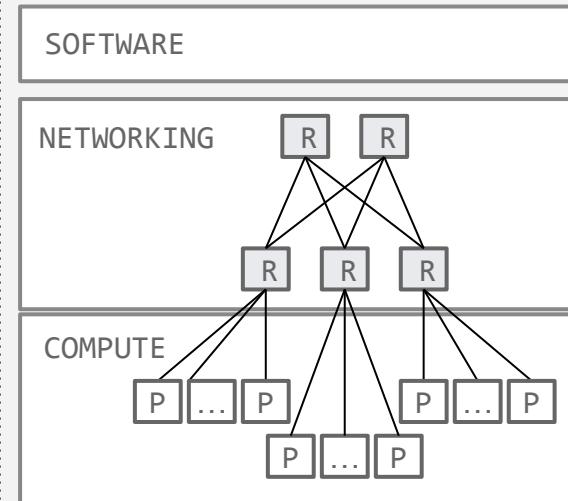
Benefits:

- Lower latency
- Higher bandwidth
- Massive scale
- Low load sensitivity

## Groq RealScale™



Disjoint Routing and Processing



# Networking

Synchronous and Statically Scheduled

Recipe:

- n LPUs
- ~16n cables
- 1 amazing software stack

Benefits:

- Lower latency
- Higher bandwidth
- Massive scale
- Low load sensitivity

**Groq RealScale™**



Predictable:  
low latency, high utilization

Disjoint Routing and Processing

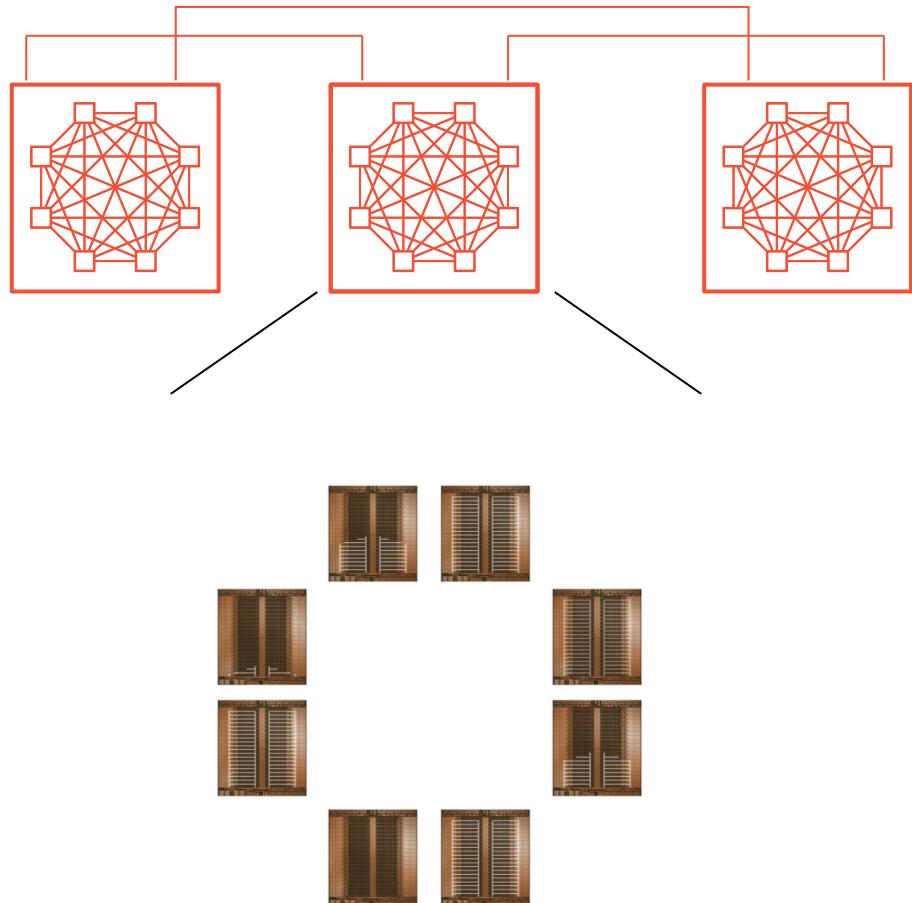


Adaptive:  
High latency, low utilization

# One Clock

Latency, Bandwidth, Scale advantages

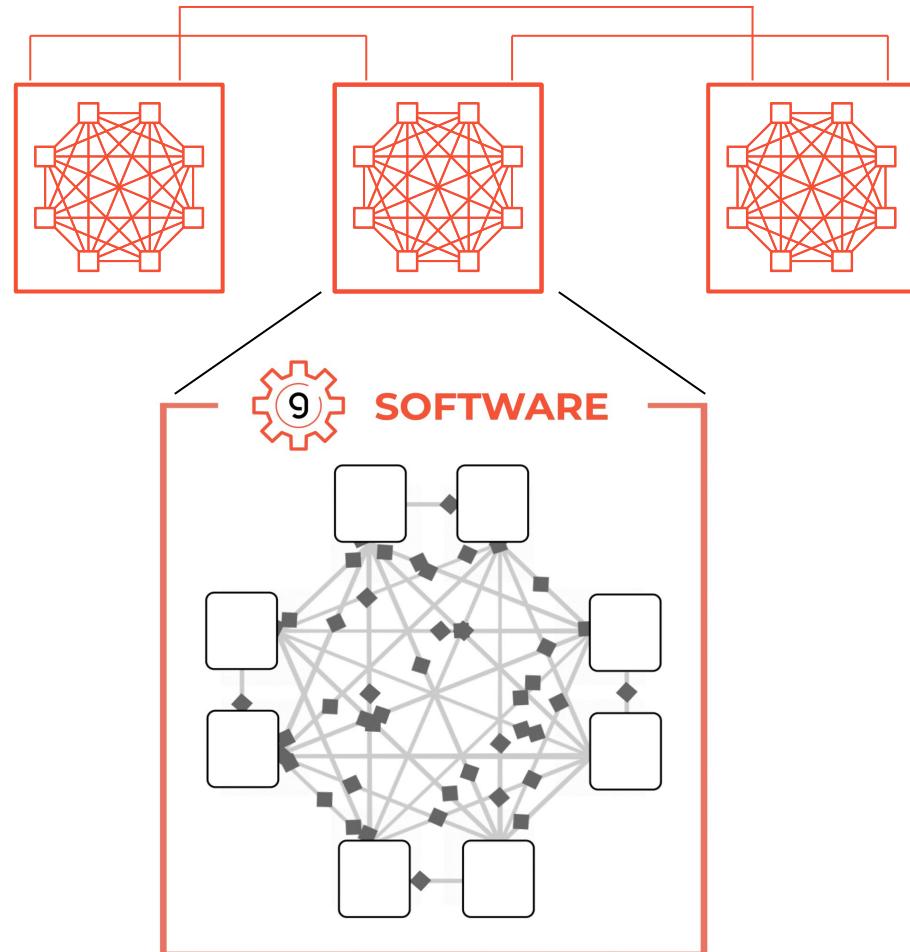
- Single core cluster
- Plesiochronous



# One Clock

Latency, Bandwidth, Scale advantages

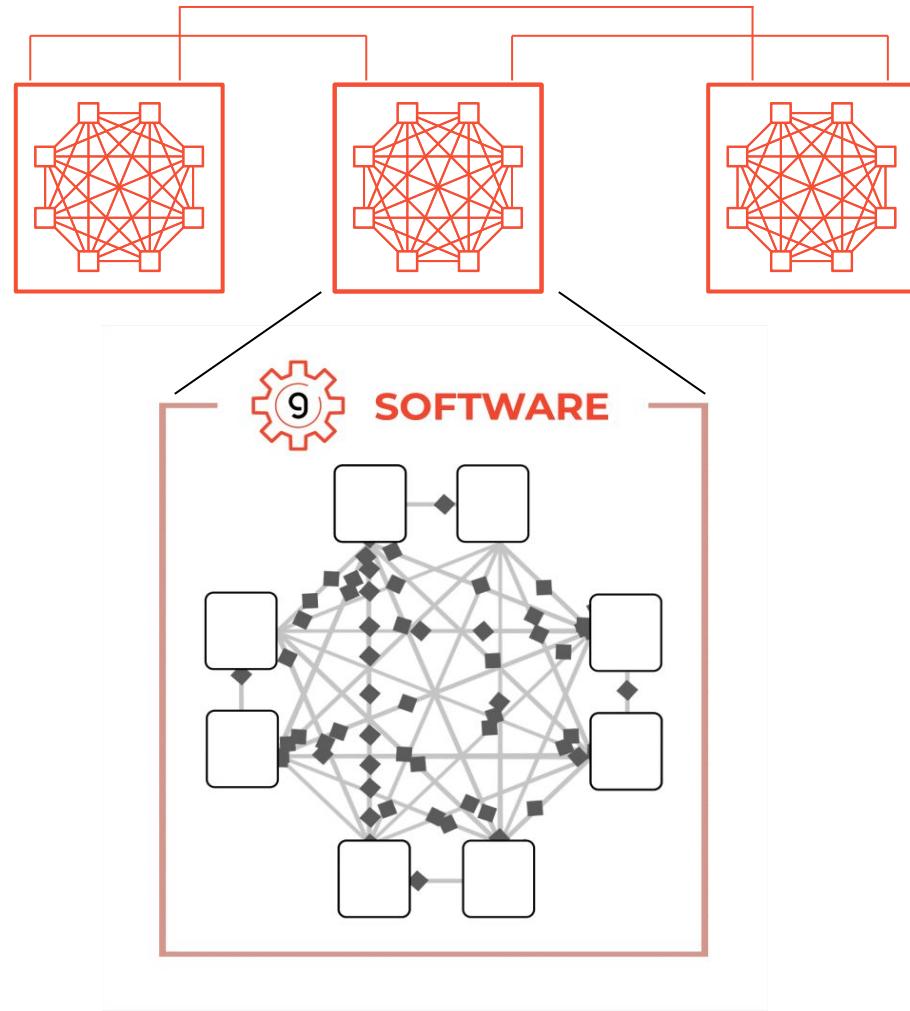
- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size



# One Clock

Latency, Bandwidth, Scale advantages

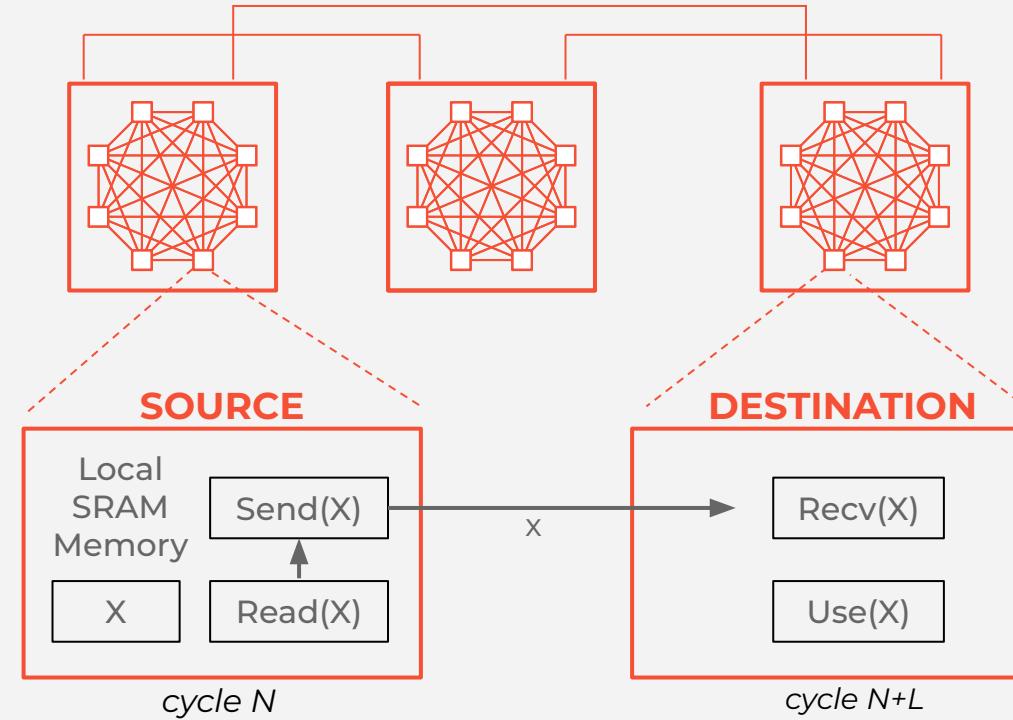
- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size
- Fun optimizations



# One Clock

Latency, Bandwidth, Scale advantages

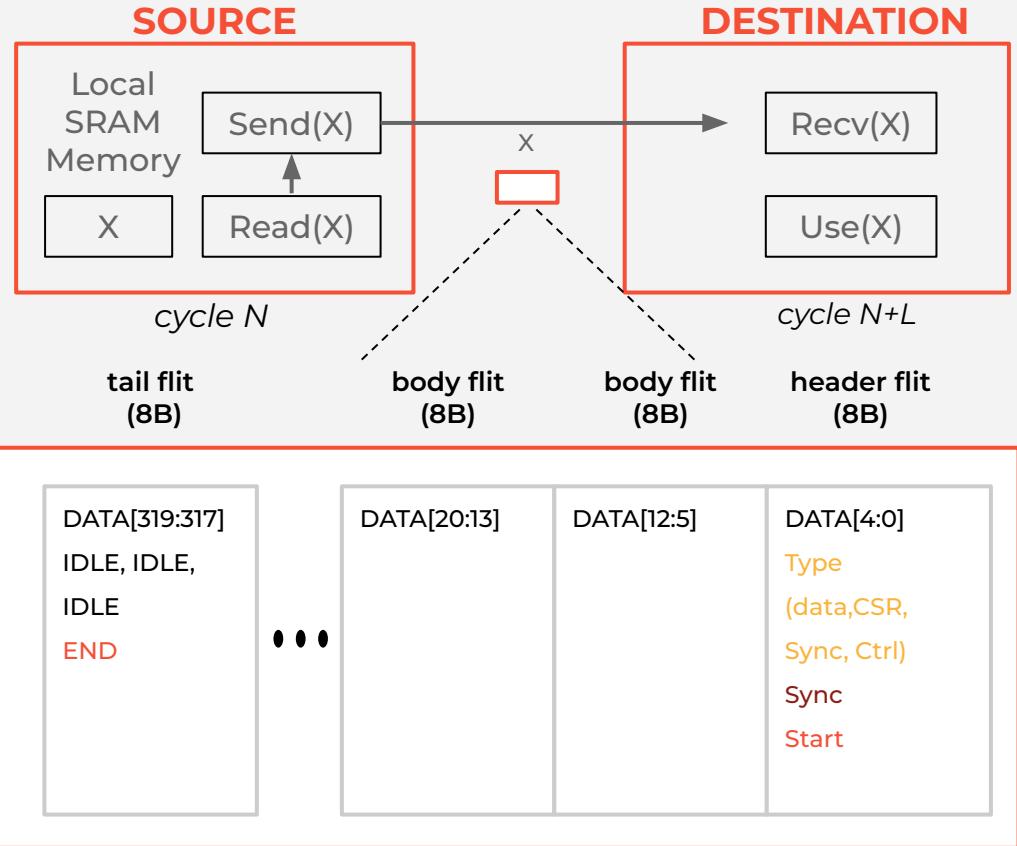
- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size
- Fun optimizations
- No messaging protocol



# One Clock

Latency, Bandwidth, Scale advantages

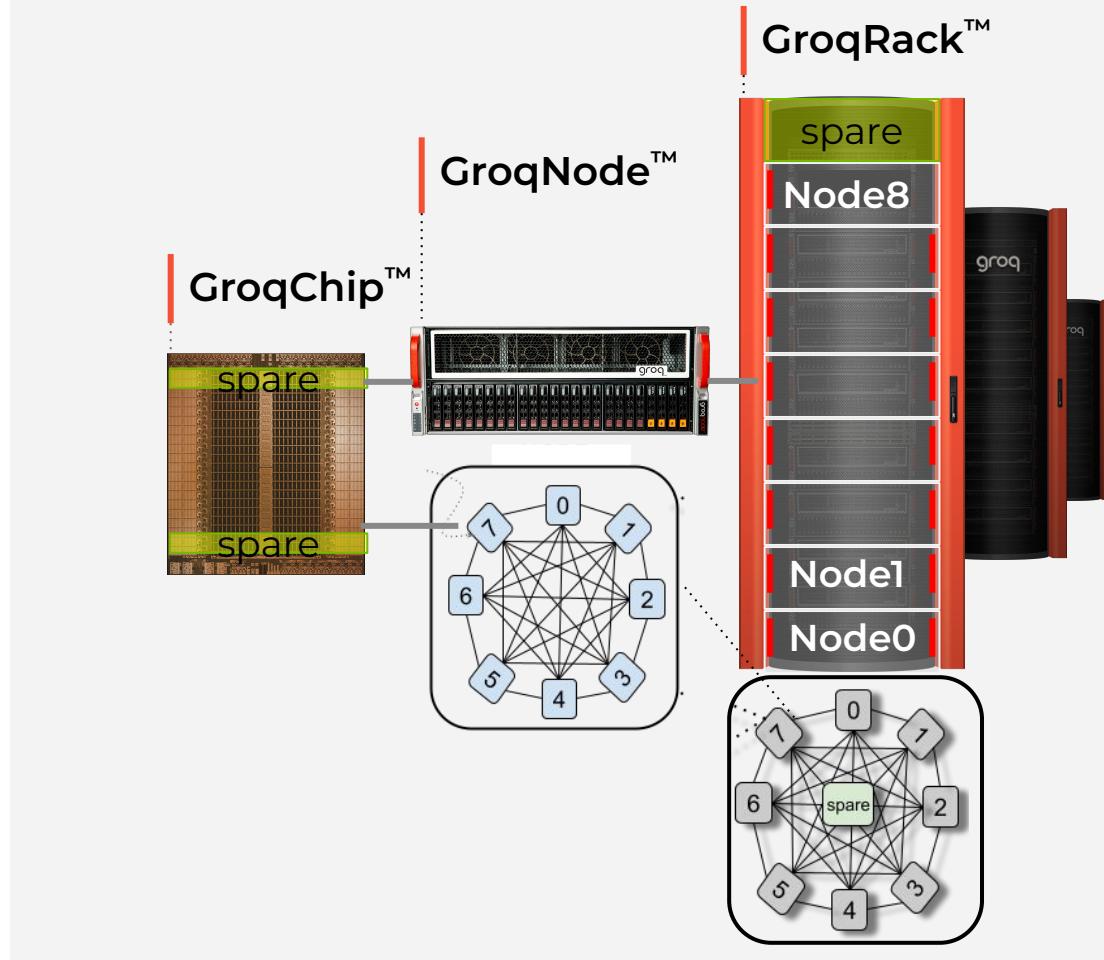
- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size
- Fun optimizations
- No messaging protocol
- Low packet overhead



# One Clock

Latency, Bandwidth, Scale advantages

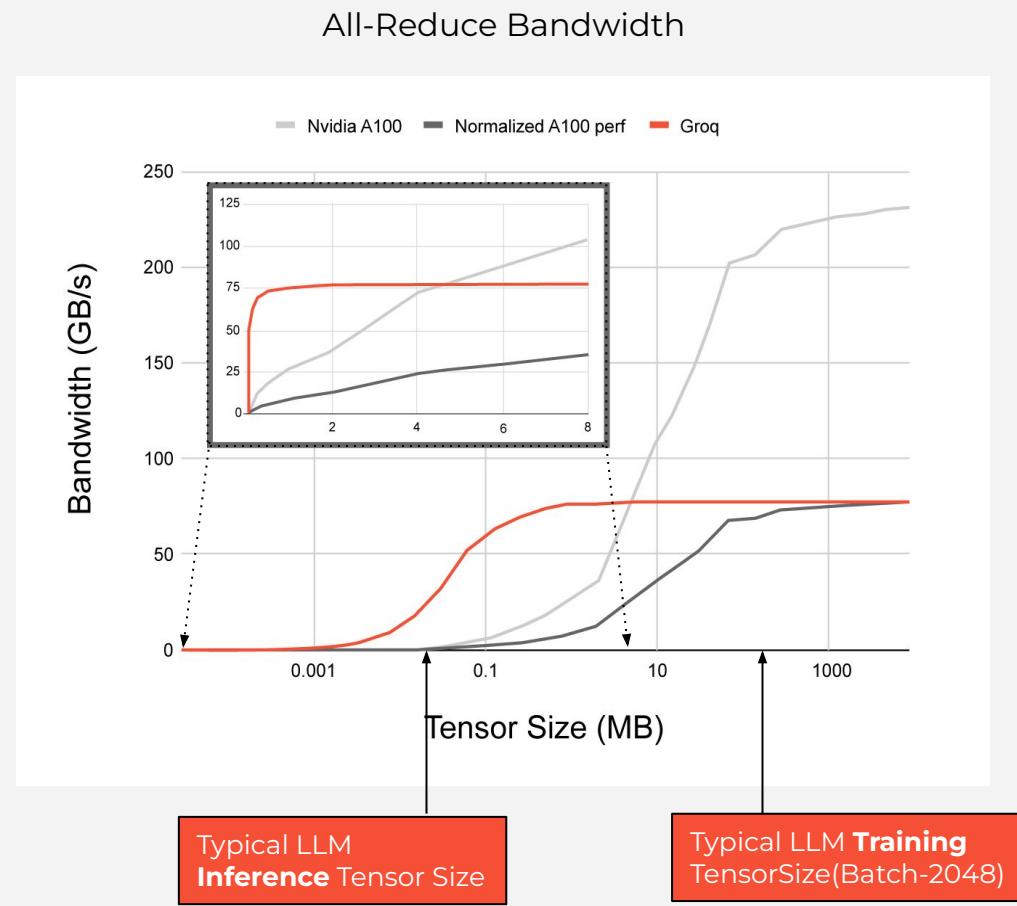
- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size
- Fun optimizations
- No messaging protocol
- Low packet overhead
- Low-Diameter topology



# One Clock

Latency, Bandwidth, Scale advantages

- Single core cluster
- Plesiochronous
- Statically routed
- Unlimited system size
- Fun optimizations
- No messaging protocol
- Low packet overhead
- Low-Diameter topology
-  Massive **networking advantage**





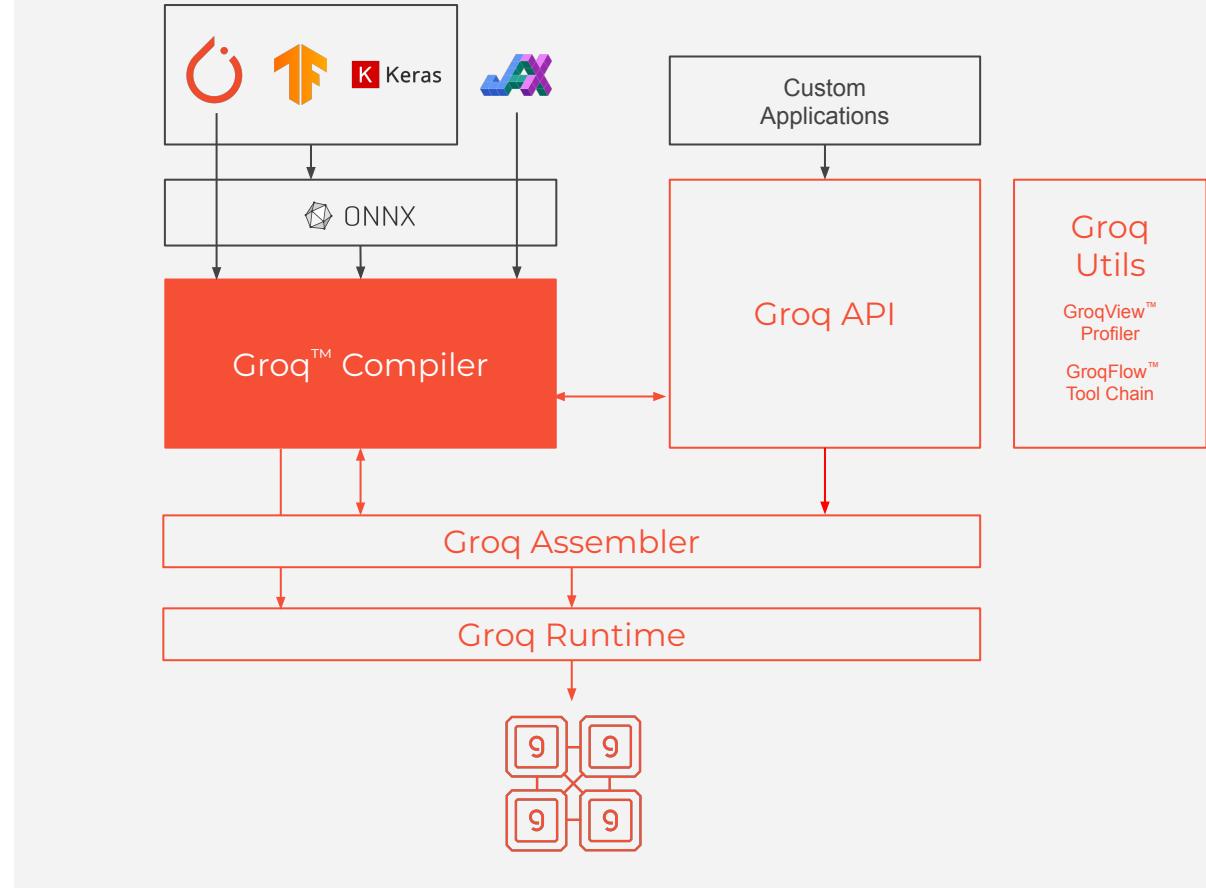
# Scaling Inference on LPUs

↑ Scaling in **Space**.

# GroqWare

Groq's Software Stack

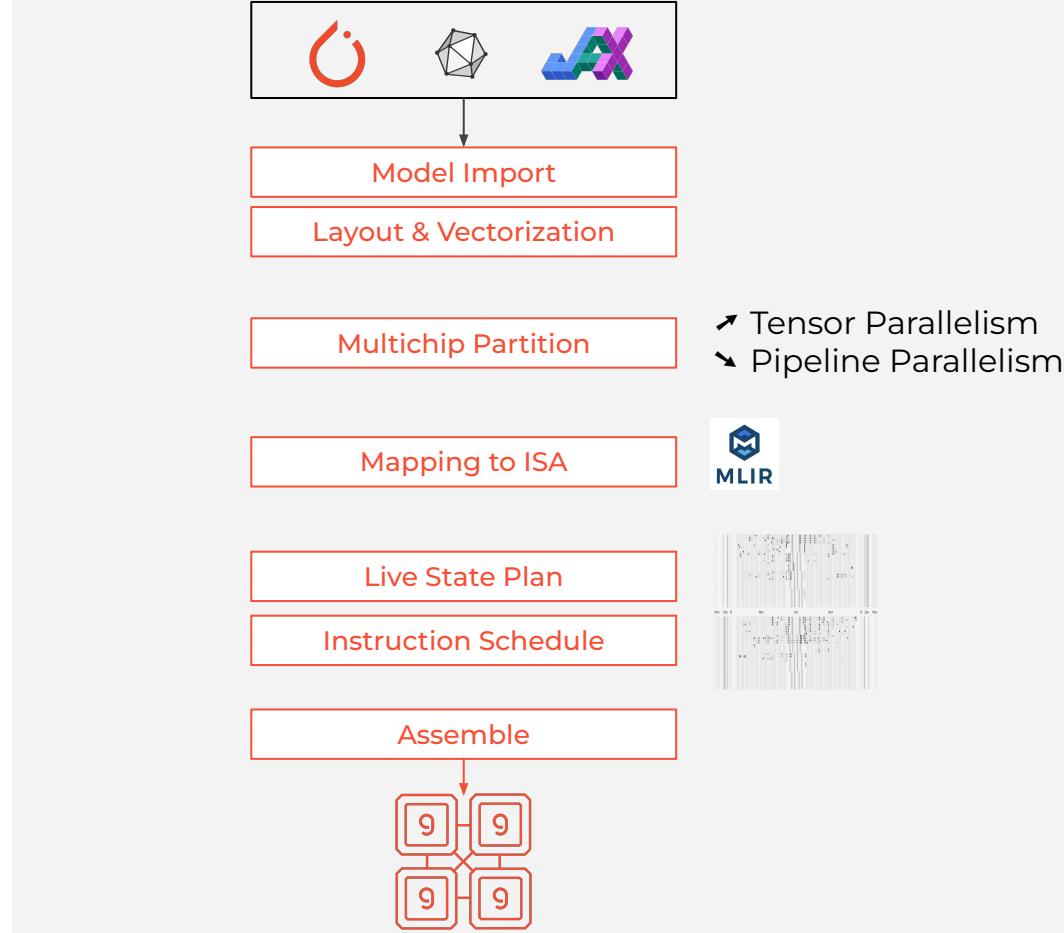
- Modular + Monolithic



# GroqWare

Groq's Software Stack

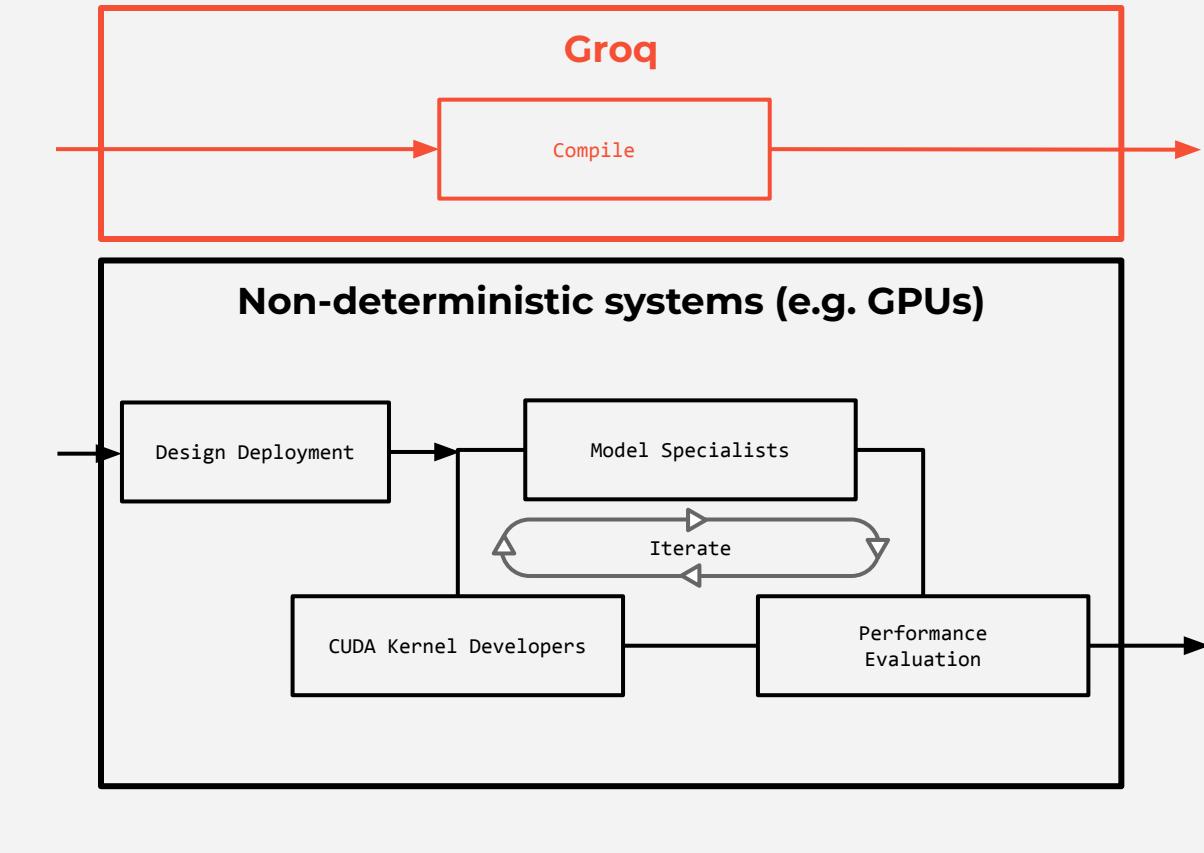
- Modular + Monolithic
- Compiler at the core



# GroqWare

Groq's Software Stack

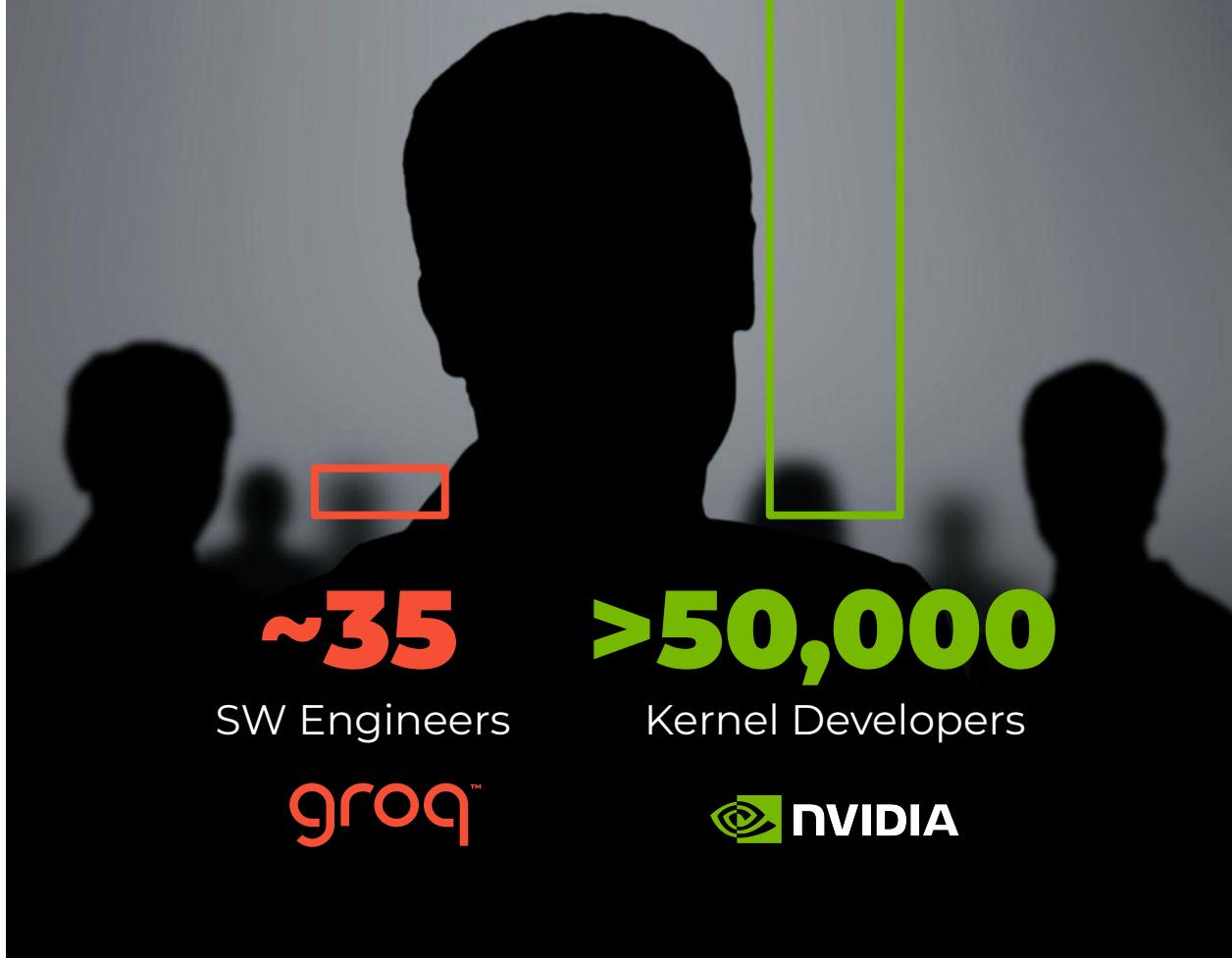
- Modular + Monolithic
- Compiler at the core
- Kernel-free



# GroqWare

Groq's Software Stack

- Modular + Monolithic
- Compiler at the core
- Kernel-free



**~35**

SW Engineers

**groq™**

**>50,000**

Kernel Developers

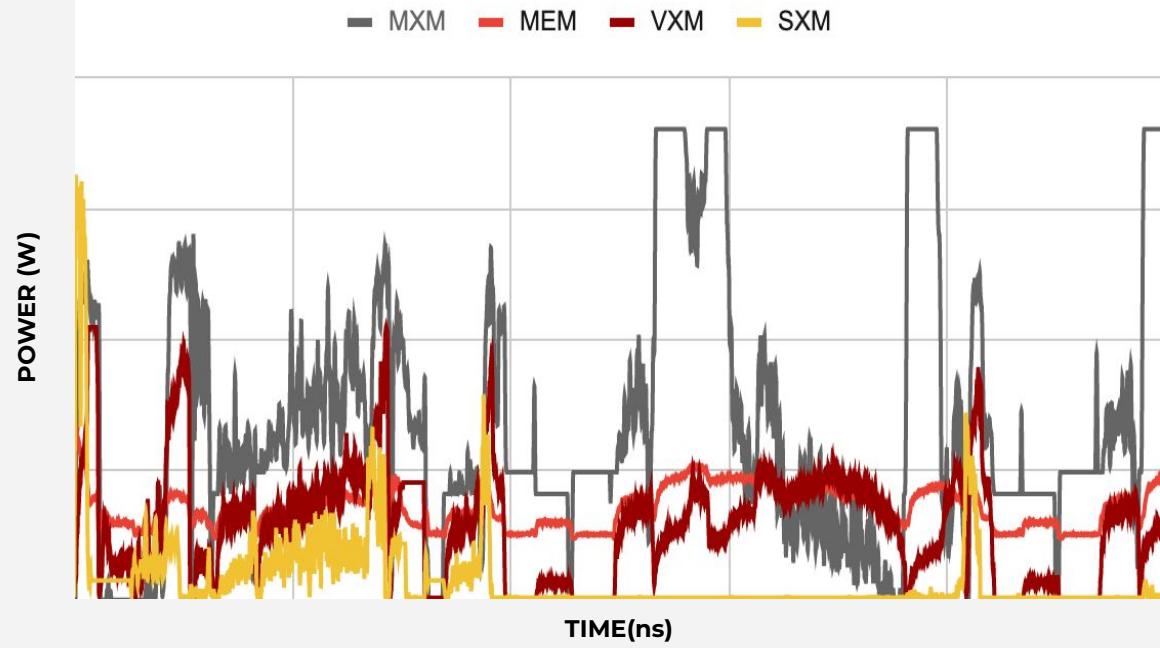
 **NVIDIA**

# GroqWare

Groq's Software Stack

- Modular + Monolithic
- Compiler at the core
- Kernel-free
- Fun optimizations

**GroqChip™ Functional Units Power Over Time**

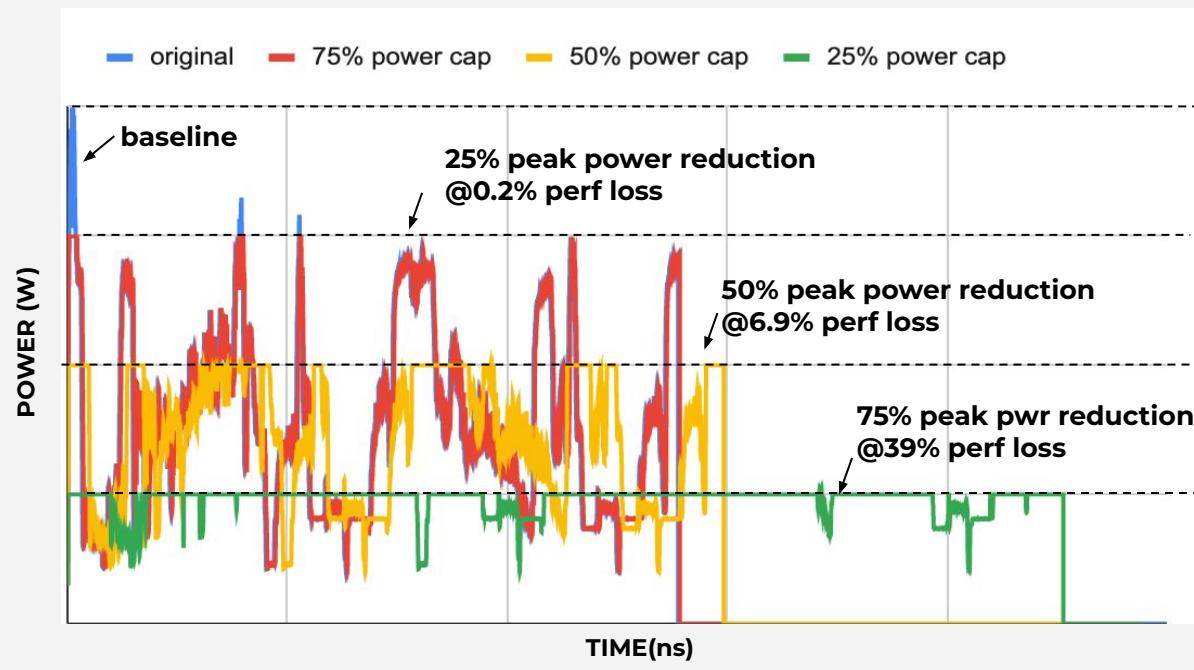


# GroqWare

Groq's Software Stack

- Modular + Monolithic
- Compiler at the core
- Kernel-free
- Fun optimizations

**GroqChip™ Total Power Over Time**



# Introducing GroqFlow™

## Step 1: Get your model

```
0 import transformers
1 import torch
2 from groqflow import groqit
3
4 model = transformers.GPT2Model(transformers.GPT2Config())
5
6 inputs = {
7     "input_ids": torch.ones(1, 1_024, dtype=torch.long),
8     "attention_mask": torch.ones(1, 1_024, dtype=torch.float),
9 }
10
11 gmodel = groqit(model,inputs)
12
13 output = gmodel(**inputs)
14
15
```

# Introducing GroqFlow™

## Step 2: Get some inputs

```
0 import transformers
1 import torch
2 from groqflow import groqit
3
4 model = transformers.GPT2Model(transformers.GPT2Config())
5
6 inputs = {
7     "input_ids": torch.ones(1, 1_024, dtype=torch.long),
8     "attention_mask": torch.ones(1, 1_024, dtype=torch.float),
9 }
10
11 gmodel = groqit(model,inputs)
12
13 output = gmodel(**inputs)
14
15
```

# Introducing GroqFlow™

## Step 3: Just Groq it!

```
0 import transformers  
1 import torch  
2 from groqflow import groqit  
3  
4 model = transformers.GPT2Model(transformers.GPT2Config())  
5  
6 inputs = {  
7     "input_ids": torch.ones(1, 1_024, dtype=torch.long),  
8     "attention_mask": torch.ones(1, 1_024, dtype=torch.float),  
9 }  
10  
11 gmodel = groqit(model,inputs) → GroqFlow is building model "bert"  
12  
13 output = gmodel(**inputs)  
14  
15
```

Converting to ONNX  
Optimizing ONNX file  
Checking for Op support  
Converting to FP16  
Compiling model  
Assembling model

# Introducing GroqFlow™

Inference is easy!

```
0 import transformers  
1 import torch  
2 from groqflow import groqit  
3  
4 model = transformers.GPT2Model(transformers.GPT2Config())  
5  
6 inputs = {  
7     "input_ids": torch.ones(1, 1_024, dtype=torch.long),  
8     "attention_mask": torch.ones(1, 1_024, dtype=torch.float),  
9 }  
10  
11 gmodel = groqit(model,inputs)  
12  
13 output = gmodel(**inputs) → tensor([ 0.3628,  0.0489,  0.2952,  0.0022,  
14                                         -0.0161,  0.3451, -0.3209,  0.0021, ...  
15 ])
```

# Introducing GroqFlow™

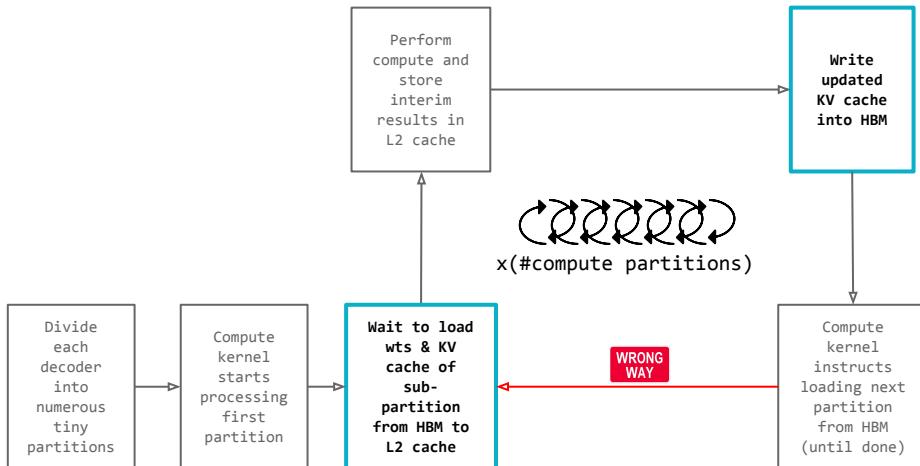
## Clear messages

What if things don't go  
as planned?

Clear feedback on how  
to move forward

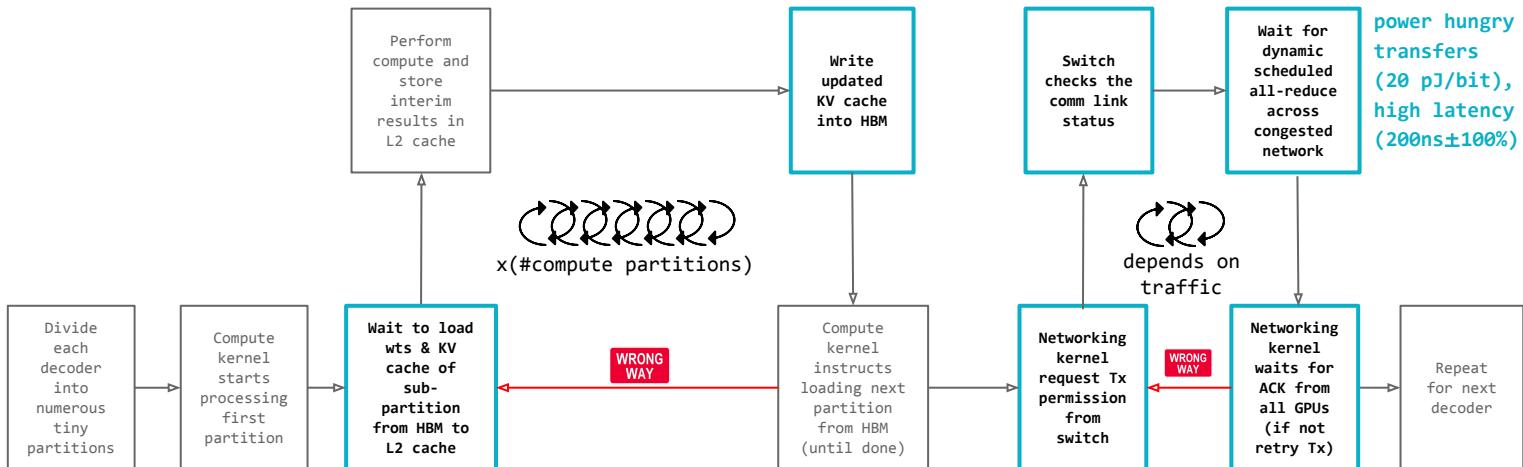
GroqFlow is building model "bert"  
Converting to ONNX  
Optimizing ONNX file  
Checking for Op support  
Converting to FP16  
Compiling model  
Assembling model

# Recall: GPU LLM Inference



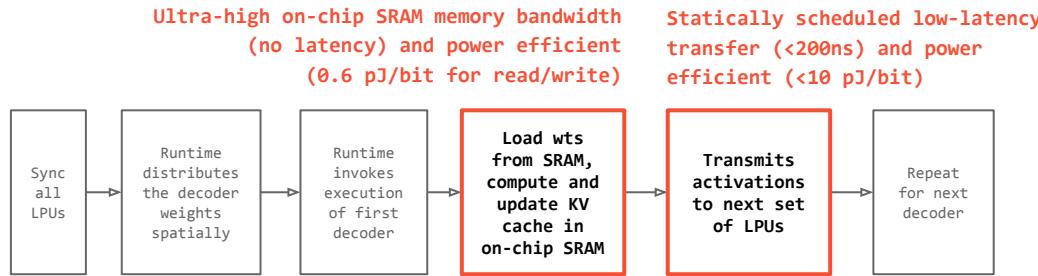
Poor off-chip HBM memory BW (high latency  
~300ns-1300ns), very high cost (HBM<sub>s</sub>/interposers),  
very power hungry (4-6 pJ/bit for read/write)

# Recall: GPU LLM Inference



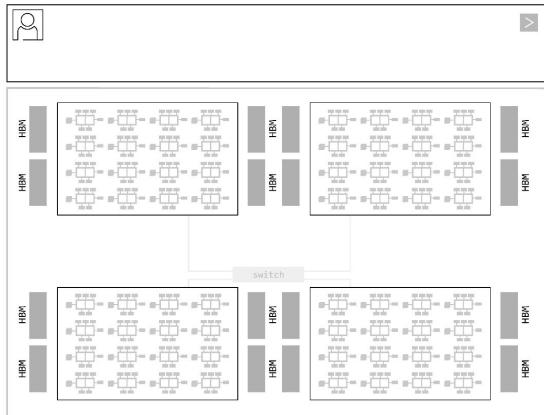
Poor off-chip HBM memory BW (high latency  
~300ns-1300ns), very high cost (HBM<sub>s</sub>/interposers),  
very power hungry (4-6 pJ/bit for read/write)

# LPU LLM Inference



# GPUs vs LPUs

## GPU Based Systems



## LPU Based Systems



# Scaling Advantages ⇒ Rate Improvements

## GPUs:

- ✖ Low HBM bandwidth **1/100X** of on-chip SRAM
- ✖ High HBM access latency (**300ns-1300ns**)
- ✖ High HBM access power (**4-6 pJ/bit for R/W**)
- ✖ Poor GPU-to-GPU collectives with asynchronous Communication

## LPU:

- ✓ 100X higher Bandwidth than HBM
- ✓ Lowest SRAM access latency (**<5 ns**)
- ✓ Lowest SRAM power (**0.3 pJ/bit for R/W**)
- ✓ Efficient LPU-to-LPU collectives due to deterministic communication

# Scaling Advantages ⇒ Rate Improvements

## GPUs:

- ✖ Low HBM bandwidth **1/100X** of on-chip SRAM
- ✖ High HBM access latency (**300ns-1300ns**)
- ✖ High HBM access power (**4-6 pJ/bit for R/W**)
- ✖ Poor GPU-to-GPU collectives with asynchronous Communication

## LPUs:

- ✓ 100X higher Bandwidth than HBM
- ✓ Lowest SRAM access latency (**<5 ns**)
- ✓ Lowest SRAM power (**0.3 pJ/bit for R/W**)
- ✓ Efficient LPU-to-LPU collectives due to deterministic communication

## Input Processing Rates (bounded by compute allocated to single user)

- ✖ Scaling challenges
- ✓ High **tensor & pipeline parallelism**

# Scaling Advantages ⇒ Rate Improvements

## GPUs:

- ✖ Low HBM bandwidth **1/100X** of on-chip SRAM
- ✖ High HBM access latency (**300ns-1300ns**)
- ✖ High HBM access power (**4-6 pJ/bit for R/W**)
- ✖ Poor GPU-to-GPU collectives with asynchronous Communication

## LPUs:

- ✓ 100X higher Bandwidth than HBM
- ✓ Lowest SRAM access latency (**<5 ns**)
- ✓ Lowest SRAM power (**0.3 pJ/bit for R/W**)
- ✓ Efficient LPU-to-LPU collectives due to deterministic communication

### **Input Processing Rates** (bounded by compute allocated to single user)

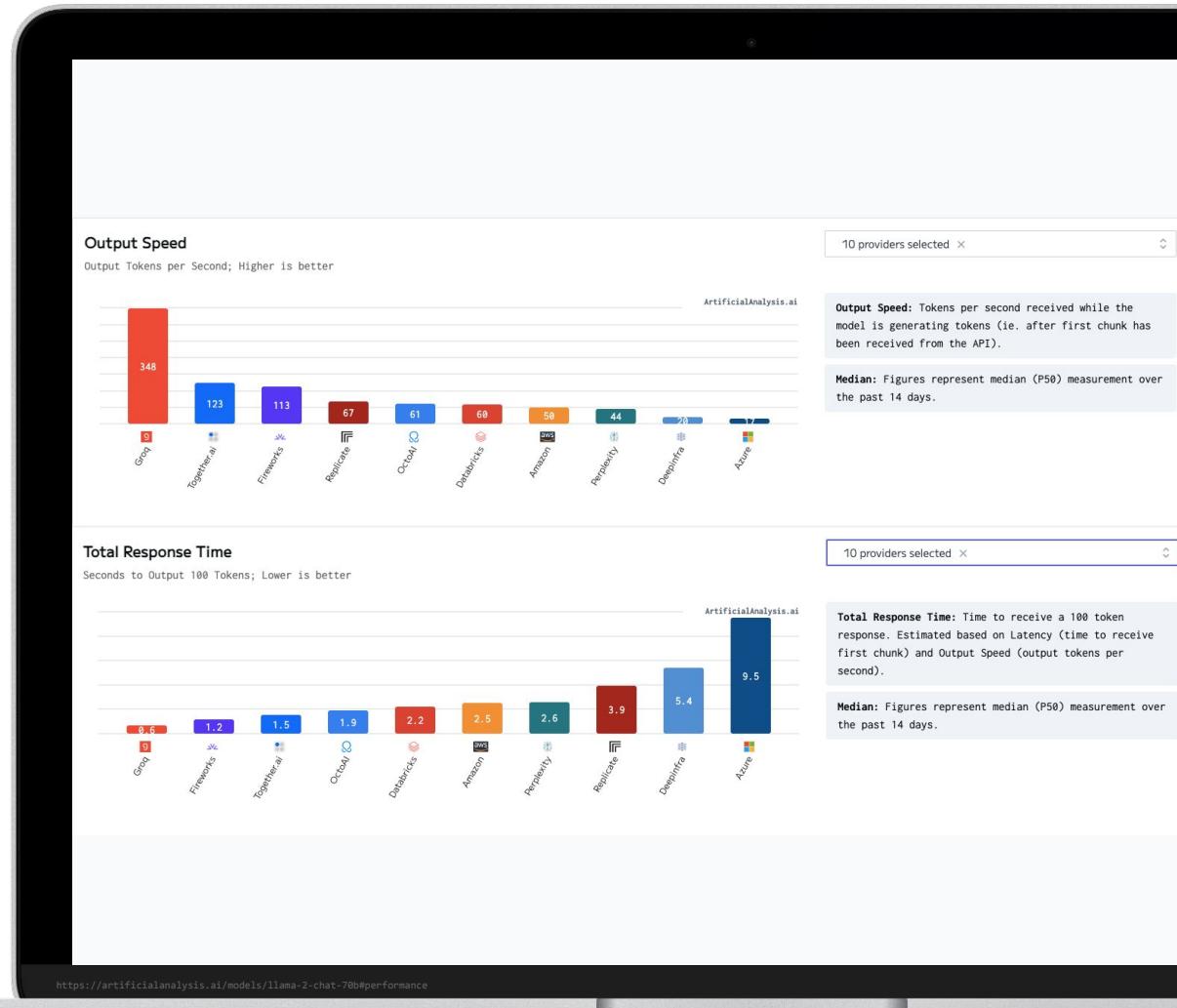
- ✖ Scaling challenges
- ✓ High **tensor & pipeline parallelism**

### **Output Generation Rates** (bounded by memory bandwidth)

- ✖ Scaling challenges
- ✖ **High-Batch (100 - 1000)** implies additional scaling challenges and has inherent rate limitations
- ✓ High **tensor parallelism**
- ✓ Leverage **Low-Batch (5-20)** approach which unlocks high rates, more efficient collectives and **deep pipeline parallelism**

# 3rd Party Benchmarks

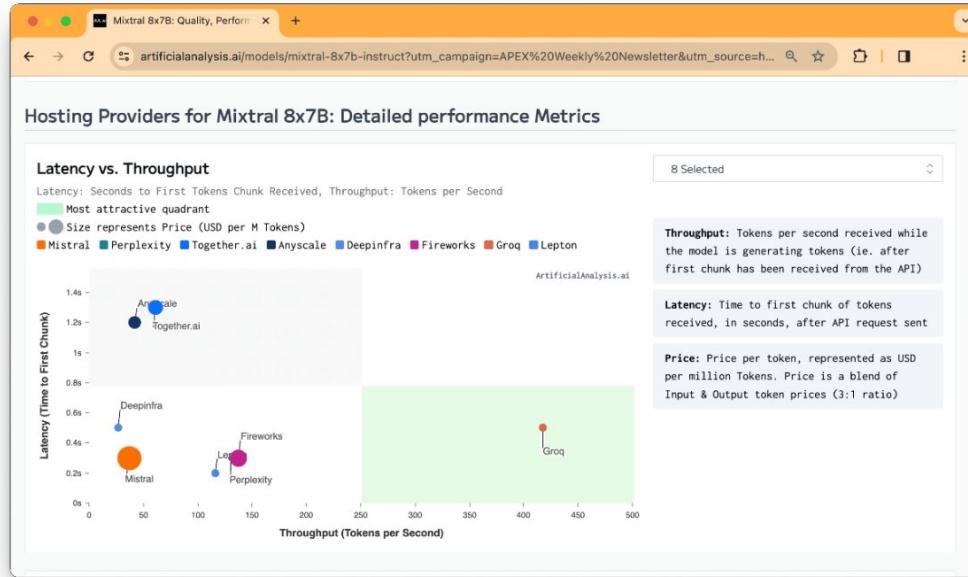
## Groundbreaking Llama 3 70B Performance



# Fastest Inference, Period.

Groq has demonstrated 15x faster LLM inference performance on an ArtificialAnalysis.ai leaderboard compared to the top cloud-based providers.

In this public benchmark, Mistral.ai's Mixtral 8x7B Instruct running on the Groq LPU™ Inference Engine outperformed all other cloud-based inference providers at up to 15x faster output tokens throughput.



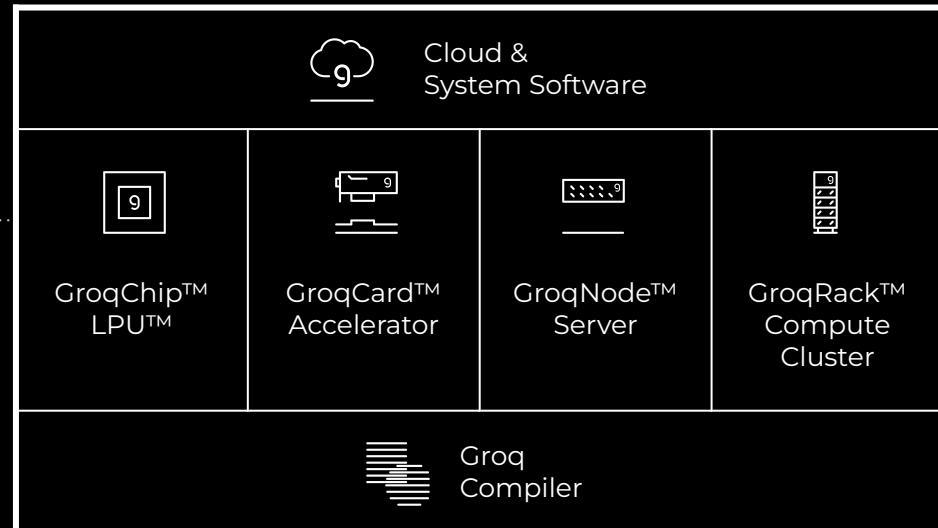
≡ AI Inference Solution

# Vertically Integrated AI Inference Solution.



## AI Inference Co-cloud

# groqcloud™



groq®

~~223K~~ 500K

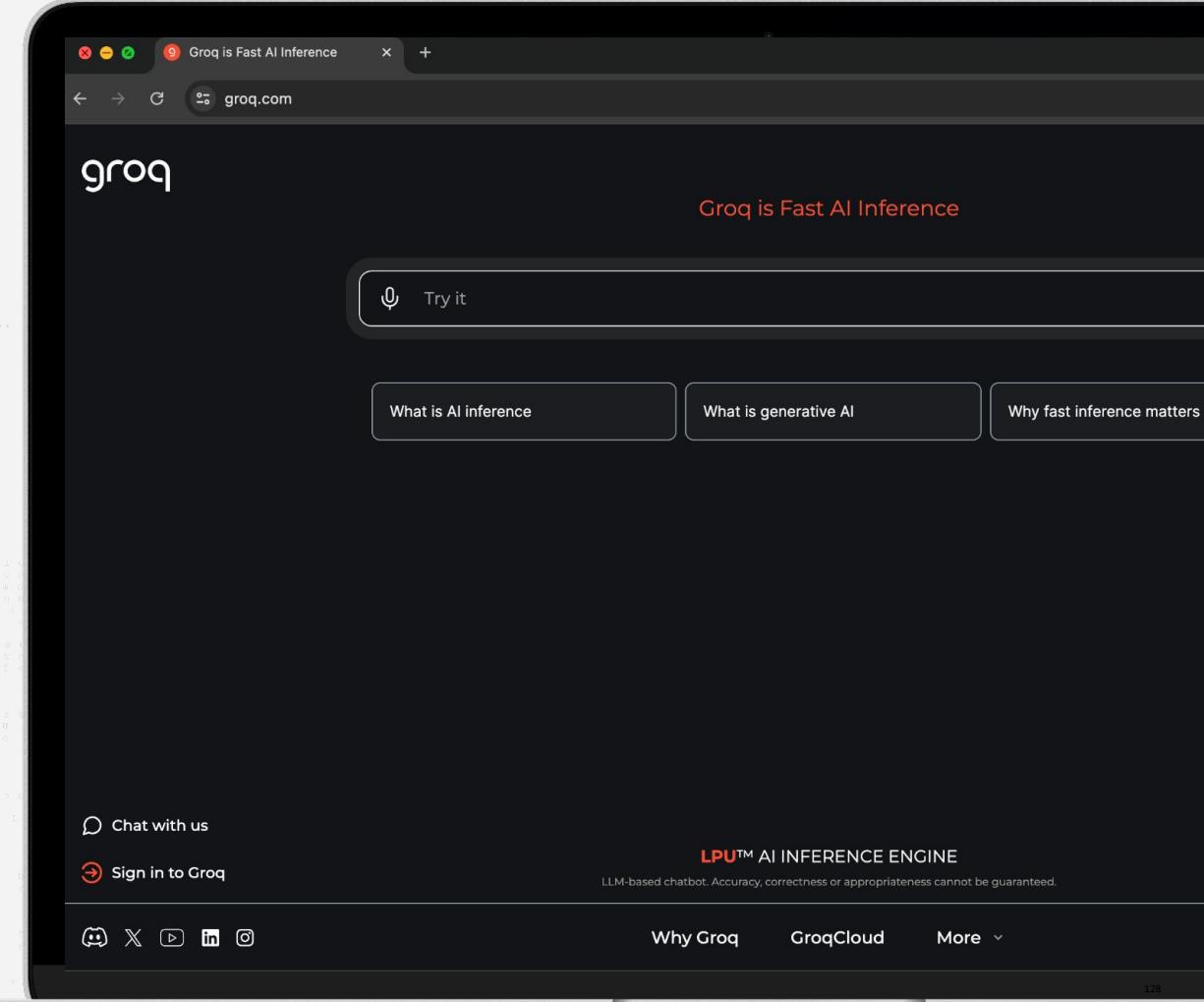
USERS >

Now developing  
on GroqCloud.



## ☰ How to get access to GroqChat?

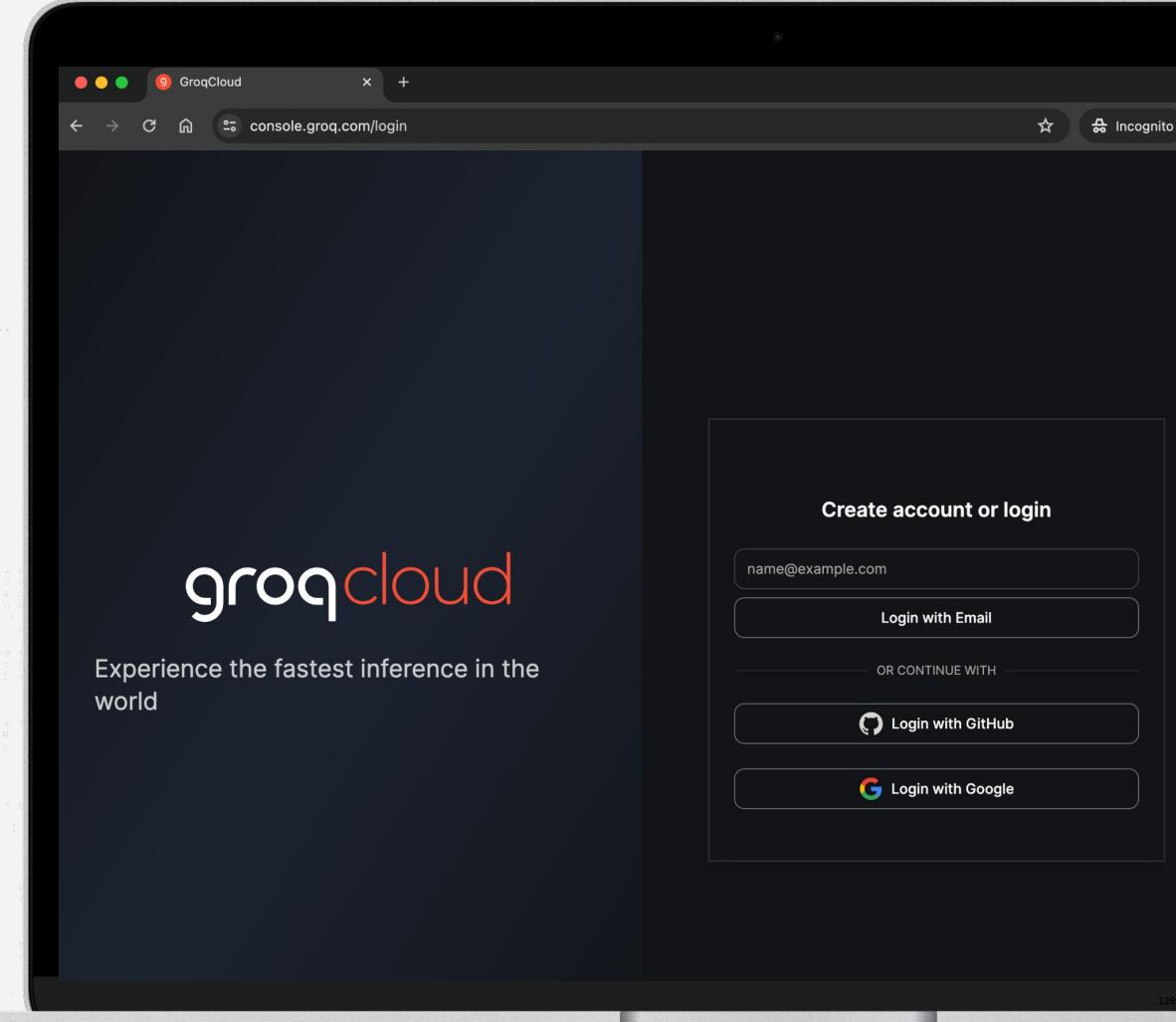
Visit  
[groq.com](https://groq.com)



☰ How to get access to Groq API?

Visit  
console.groq.com

groq®



# Dev Spotlight

A screenshot of a mobile application interface. At the top, there's a navigation bar with icons for back, forward, search, and refresh. Below it is a header section with a profile picture of Felipe Schieber, his name (@FelipeSchieber), and the date (Feb 25). The main content area features a large circular icon containing a portrait of Felipe Schieber, set against a background of a green, rolling landscape under a blue sky with clouds. To the left of the portrait is a text block from Felipe:

Exploring the concept of a generative hierarchical Wikipedia-like app powered by @GroqlInc for blazingly fast article generation.

The idea is that any person can enter a topic, their goals and background and the app will create a learning tree and write all the content.

Let me...  
Show more

At the bottom of the screen, there's a footer bar with icons for back, forward, and search.



## DEMO: Structuring Generated Content

A screenshot of a video player interface. The video content is an article titled "CMOS Technology: A Brief History and Overview". The article discusses the development of CMOS technology from the 1960s to the present, highlighting its impact on computing and digital devices. The video player includes a progress bar at the bottom showing 0:21 / 1:07, volume controls, and other standard video player icons.

Consilium  
@FelipeSchieber



# Project Media QA

Experience ultra-accelerated video and audio transcription, summarization, & QA made possible by combining open-source LLMs and ASR models both powered by Groq.

Language                    Groq Supported ASR Models                    Groq Supported LLMs  
Automatic Language Detection      Whisper V3 Large      Llama-3-8B-8192

## DEMO: Interacting with Media

Transcribe

Project Media QA  
By GroqLabs

## ☰ Groq API Cookbook

github.com/groq/  
groq-api-cookboo  
k



GitHub - groq/groq-api-cookbook

github.com/groq/groq-api-cookbook

groq/groq-api-cookbook Public

Code Issues 1 Pull requests 2 Actions Projects Security Insights

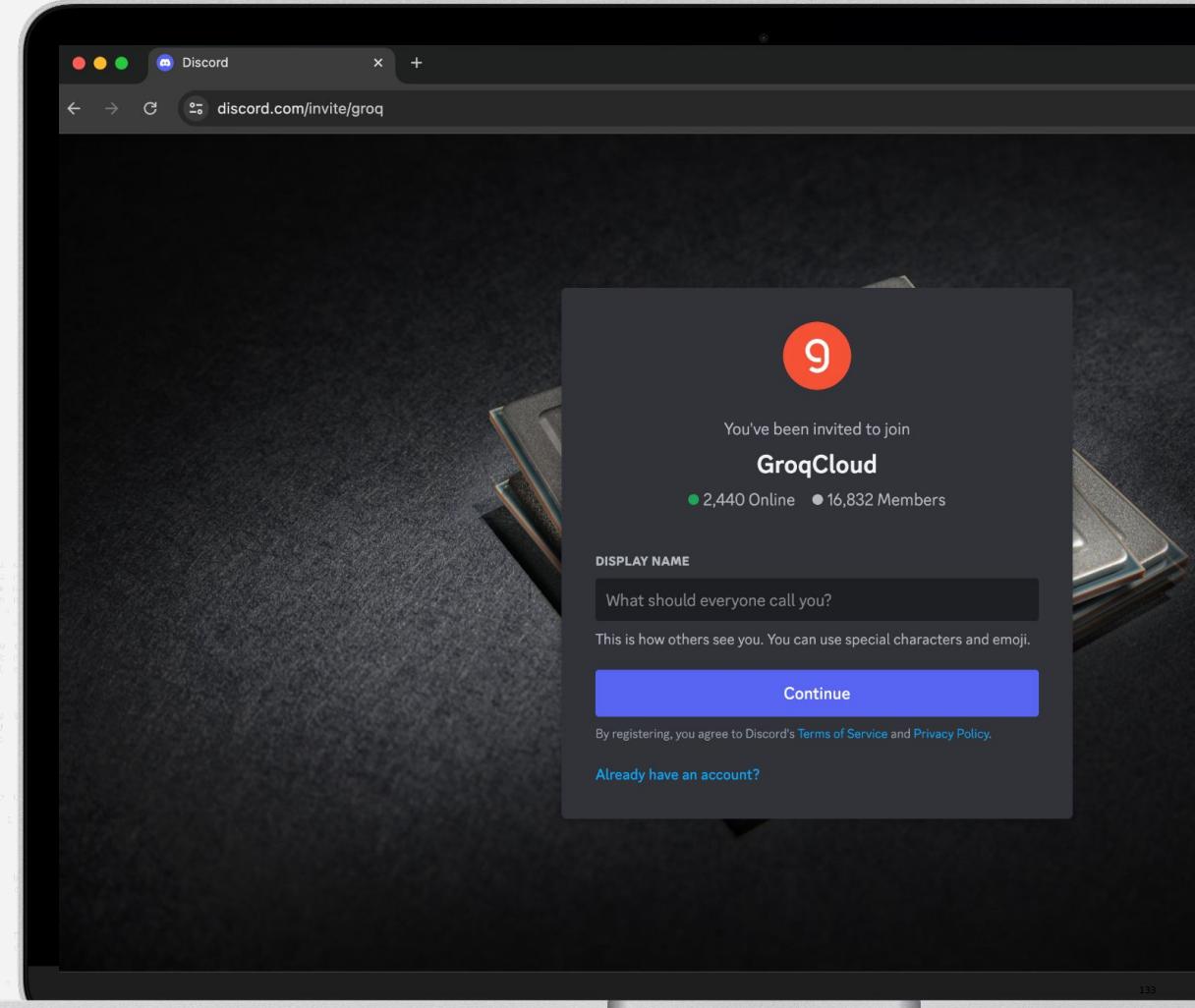
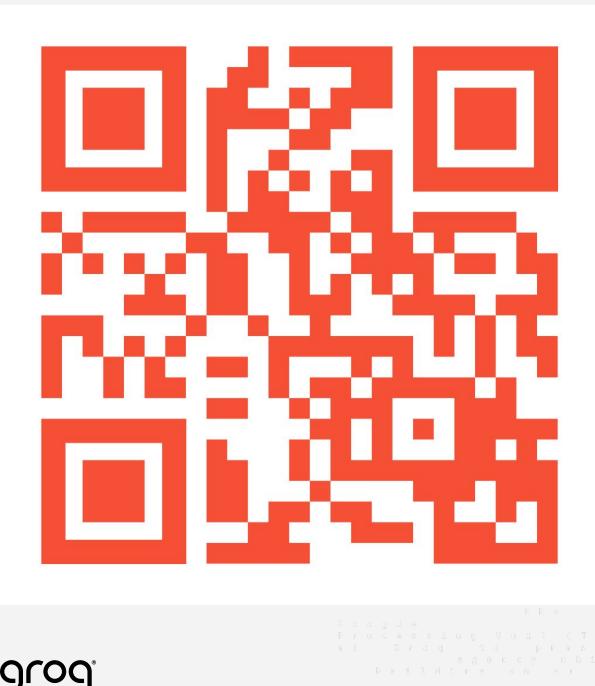
main 10 Branches 0 Tags Go to file Code

hozen-groq Merge pull request #12 from langroid/main 2ba789e · 2 weeks ago 62 Commits

File	Description	Time
benchmarking-rag-langchain	add a conclusion, slight change to docs	last month
function-calling-101-commerce	fix: enhancements to cookbook posts	last month
function-calling-sql	fix: enhancements to cookbook posts	last month
groq_streamlit_demo	Update streamlit_app.py	3 months ago
images	add logo	3 months ago
json-mode-social-determinants-of-health	fix: enhancements to cookbook posts	last month
langroid-lm-agents	update nb	2 weeks ago
llama3-stock-market-function-calling	use static png plot so that it shows up on GH preview	3 months ago
parallel-tool-use	docs: more comments	2 months ago
presidential-speeches-rag	fix: enhancements to cookbook posts	last month
replit-examples	update replit templates	2 months ago
whisper-podcast-rag	whisper cookbook entry	3 months ago
.gitignore	feat: parallel tool use example	2 months ago
CONTRIBUTING.md	Create CONTRIBUTING.md	4 months ago

☰ Groq Developer Community

Connect with  
Groq & 16K+ Devs



## Showcase your work



Groq Labs - Groq

www.groq.com/groq-labs/

# groq

Why Groq  
About Us  
Enterprise Access  
GroqCloud  
GroqLabs

## GroqLabs

### Make it Real

GroqLabs reduces the imagination gap. Check out the cool stuff we're experimenting with using Groq technology. **#betterongroq**

The art of the possible requires imagination, innovation, and creativity. At Groq, our team is always exploring unique ways to take advantage of our ultra-low latency value proposition. #GroqSpeed is great not only for chat but much more. It's where language and linear computation brings real-time AI to life. We're partnering with multiple internal and external stakeholders to create compelling use cases regarding audio, speech, image manipulation, video, scientific research, coding, and much more. To be clear, GroqLabs doesn't guarantee any of these projects will turn into products, but we don't guarantee they won't either! This is an opportunity to inspire and explore our on-going quest to "make it real."

**Project Know-It-All**  
Increase the knowledge of LLMs with contextual information via RAG integration.

[WATCH VIDEO](#)

**Project Front Page**  
Get the latest Financial (and other) news with LLMs enabled with real-time contextual knowledge.

[LAUNCH DEMO](#)

**Project StyleClip**  
Manipulate 1024x1024 portraits into 8 different styles simultaneously in ~0.18s.

[WATCH VIDEO](#)

groq®

134



The Groq logo is centered on a background composed of a grid of gray squares of varying shades. The logo itself is the word "groq" in a bold, black, sans-serif font. A small registered trademark symbol (®) is positioned at the top right of the letter "q".

groq®

# groq®

## Questions?

GroqCloud™



API / Dev Tools

GroqChat™

