

# Getting Started with Groq



# Hello Groq ...



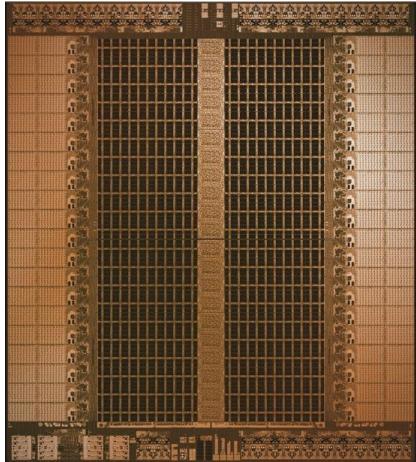
Sanjif, Software Engineer



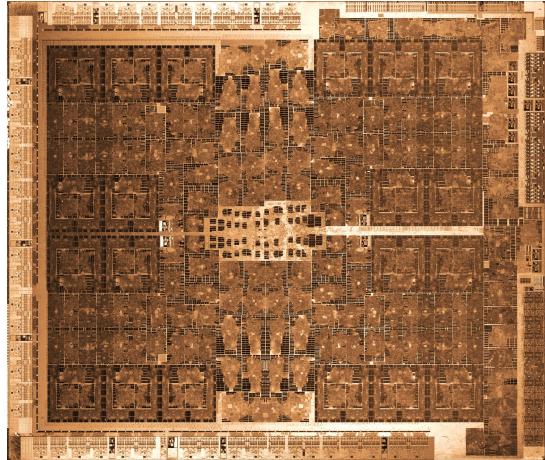
# GroqChip™ Architecture

Simply  
Better

Language Processing Unit™  
Accelerator  
GroqChip™ 1

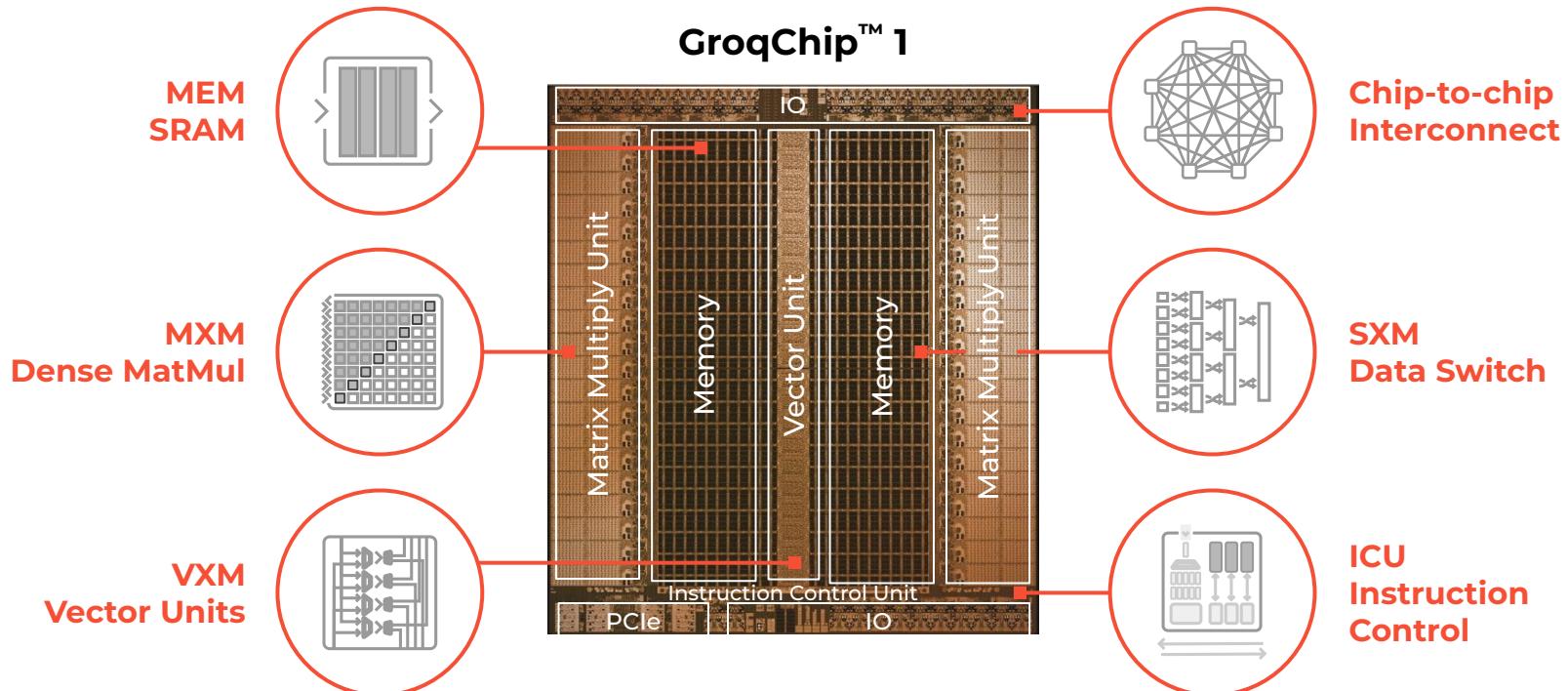


Typical Graphics  
Processor  
(GPU)



# GroqChip™ At-a-glance

Scalable compute architecture



# GroqChip™ 1 Overview

Scalable compute architecture

## SRAM Memory

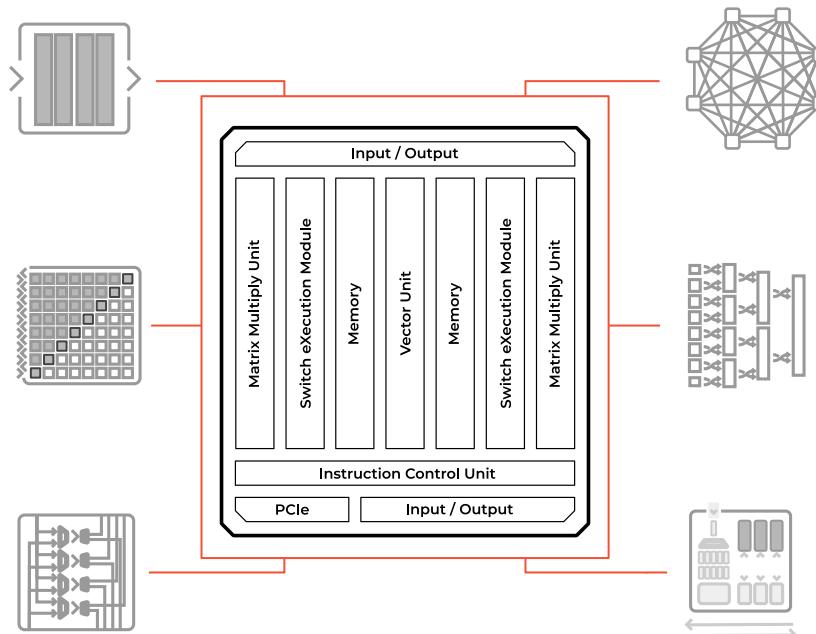
Massive concurrency  
80 TB/s of BW  
220MB capacity  
Stride insensitive

## Groq TruePoint™ Matrix

4x Engines  
750 TOP/s int8  
188 TFLOP/s fp16  
320x320 fused dot product

## Programmable Vector Units

5,120 Vector ALUs for high performance



## Networking

480 GB/s bandwidth  
Extensible network scalability  
Multiple topologies

## Data Switch

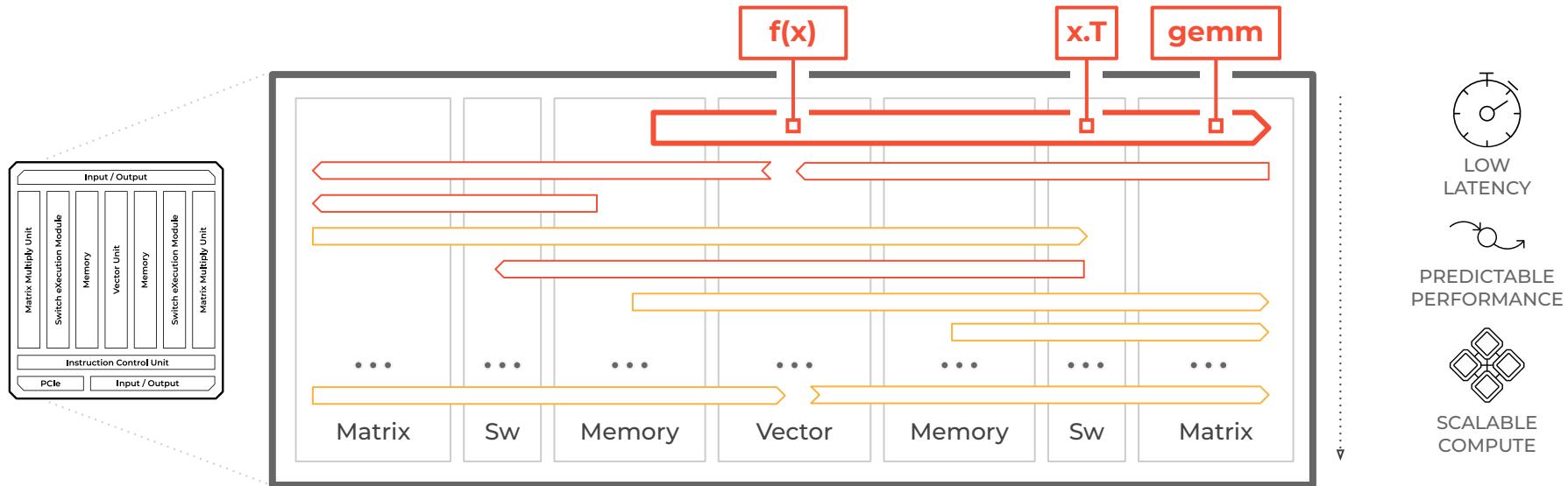
Shift, Transpose, Permuter for improved data movement and data reshapes

## Instruction Control

Multiple instruction queues for instruction parallelism

# Radically Simplifying Compute

Disaggregation and spatial processing, a simply efficient approach to SW/HW architecture



Single core,  
spatial pipeline  
processing fabric

Simple tensor  
instruction  
set arch

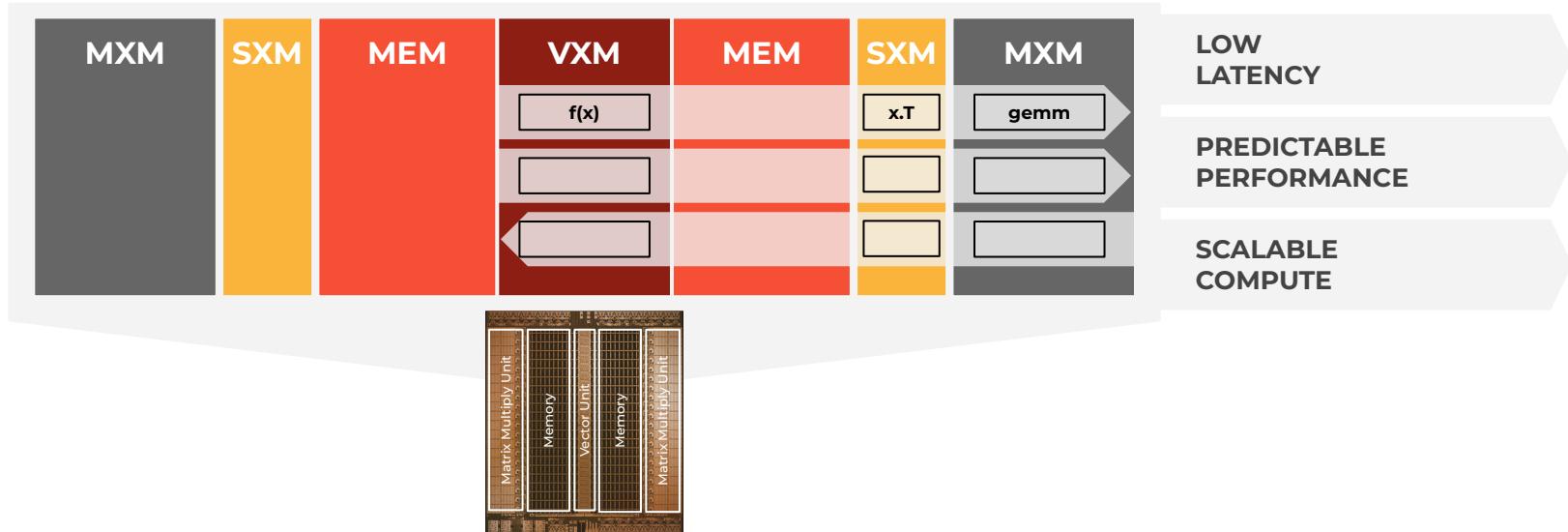
Large  
on-chip  
memory bandwidth

Deterministic,  
predictable performance  
scales to multi-chip

Stream programming  
of massive SIMD,  
concurrent streams

# Tensor Streaming Dataflow

A simply efficient approach to compute architecture and data flow



Single core,  
spatial pipeline processing

Simple tensor instruction  
set architecture

Stream programming  
of massive SIMD,  
concurrent streams

Large on-chip  
memory  
bandwidth

Deterministic, predictable  
performance scales to  
multi-chip

# Compute, Memory, and IO

## Compute

Matmult TOPs: INT8

Hemispheres \*

Planes/Hemisphere \*  $(2^{VL} + 1)$  \*

MXM width \* clock

$$2 * 2 * (2^{320} + 1) * 320 * \text{fclk} \\ = 1024 \text{ TOps}$$

(1 Peta Op per Second @1.25GHz)

## Memory

Memory = Slices/hemisphere \*  
Hemispheres \* Addresses/Bank \*  
Banks/slice \* Bytes/address

$$44 * 2 * 4096 * 2 * 320 = 220\text{MB}$$

## Memory-Compute Units BW

BW = Number of streams in parallel in both directions \*  
Bytes/stream \* Clock \*  
hemispheres \* Computation units  
 $= 64 * 320 * 900 \text{ MHz} * 2 * 2 = 80\text{TB/s}$

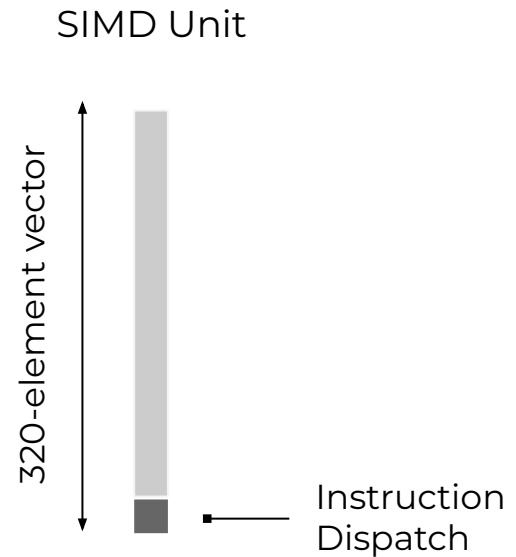
## IO BW

PCI Gen 4: x16: 32GB/s

## RealScale™ BW

16 x4 links @ 30Gb/s x 2 (bidi) =  
3.84 Tb/s = 480 GB/s of total off chip BW

# GroqChip™ Building Blocks



# GroqChip™ Building Blocks

Build different types of specialized SIMD units



**MXM**  
Matrix-Vector/  
Matrix-Matrix Multiply



**VXM**  
Vector-Vector  
Operations



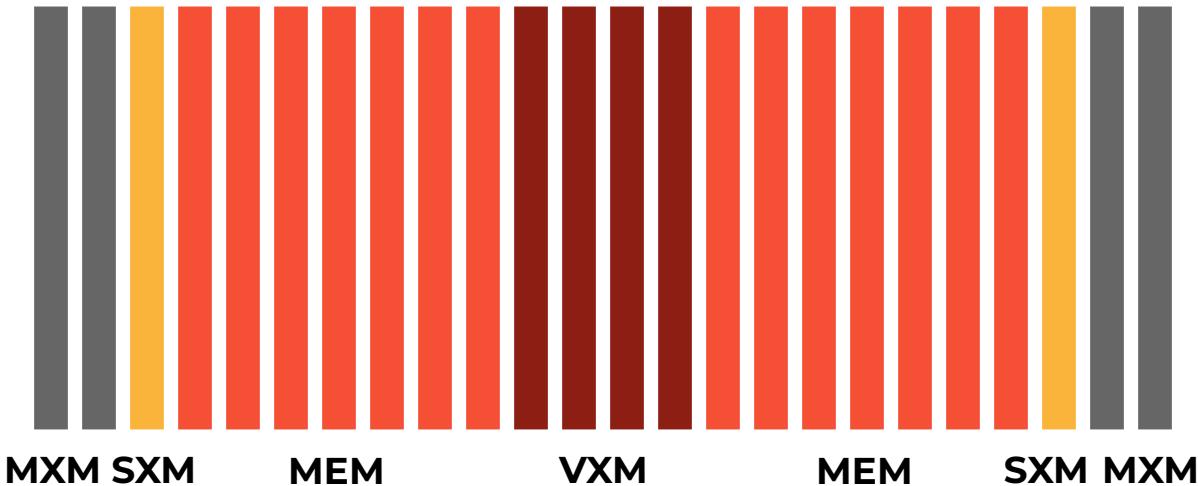
**SXM**  
Data Reshapes



**MEM**  
On-chip SRAM

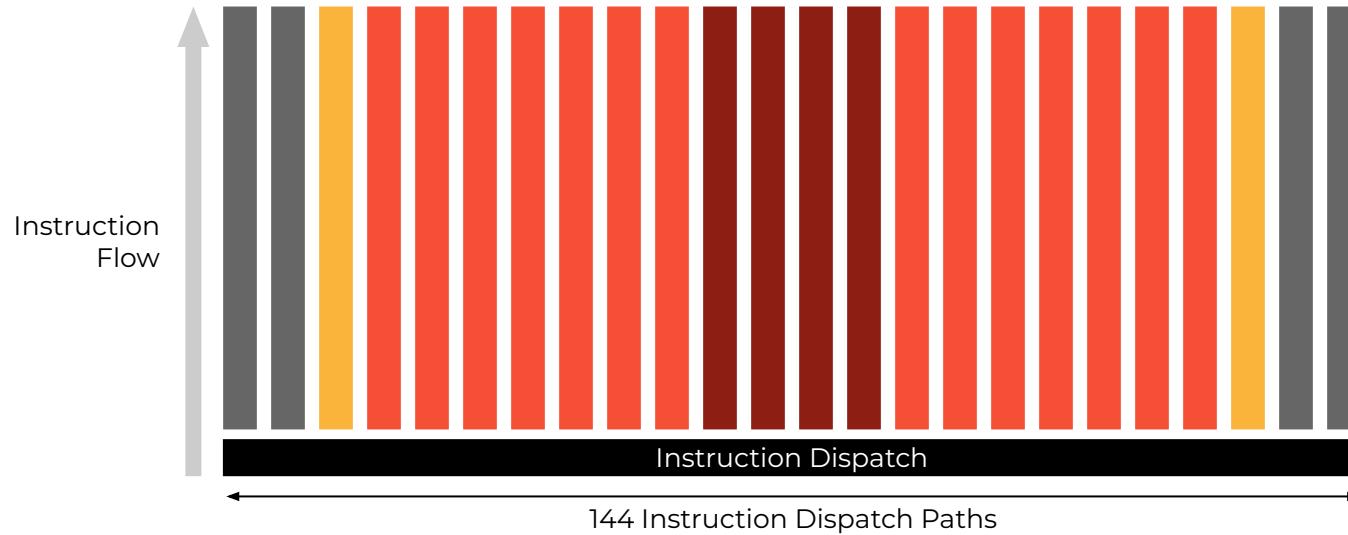
# GroqChip™ Building Blocks

Lay out SIMD units across chip area



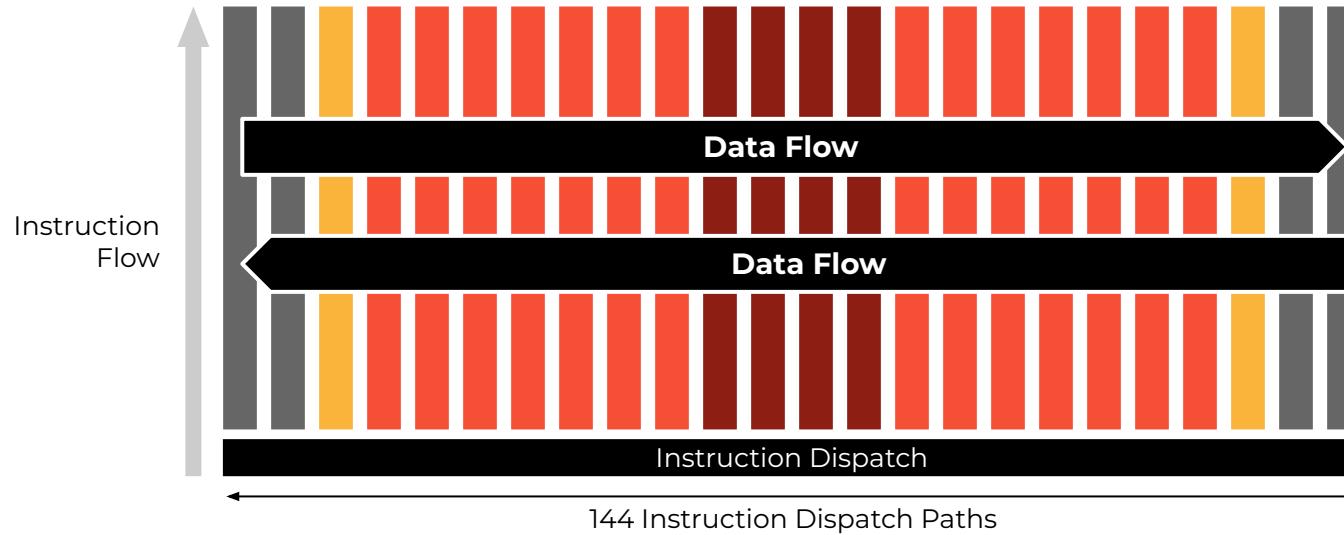
# GroqChip™ Building Blocks

Synchronized instruction dispatch across all SIMD units for lockstep execution



# GroqChip™ Building Blocks

High-bandwidth “Stream Registers” for passing data between units



# An ISA That Empowers Software

**Software-controlled memory  
enabled by low-level abstraction  
exposed by architecture**

No dynamic hardware caching

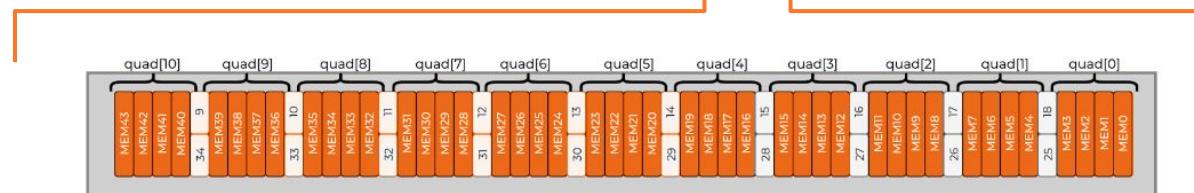
- Compiler aware of all data locations at any given point in time

Flat memory hierarchy (no L1, L2, L3, etc)

- Memory exposed to software as a set of physical banks that are directly addressed

Large on-chip memory capacity (220 MB) at high-bandwidth with (55TBps) reduces need to spill to non-deterministic DRAM

- Provides enough “scratchpad” memory to hide external memory accesses behind compute



# An ISA That Empowers Software

## Compiler empowered to perform Cycle-accurate instruction scheduling

Functional units execute in lockstep

- One instruction issued per cycle at each dispatch path
  - Can be viewed as fully-pipelined 144-wide VLIW instructions

Little hardware control needed for managing instruction execution

- < 3% area overhead for instruction dispatch logic



# An ISA That Empowers Software

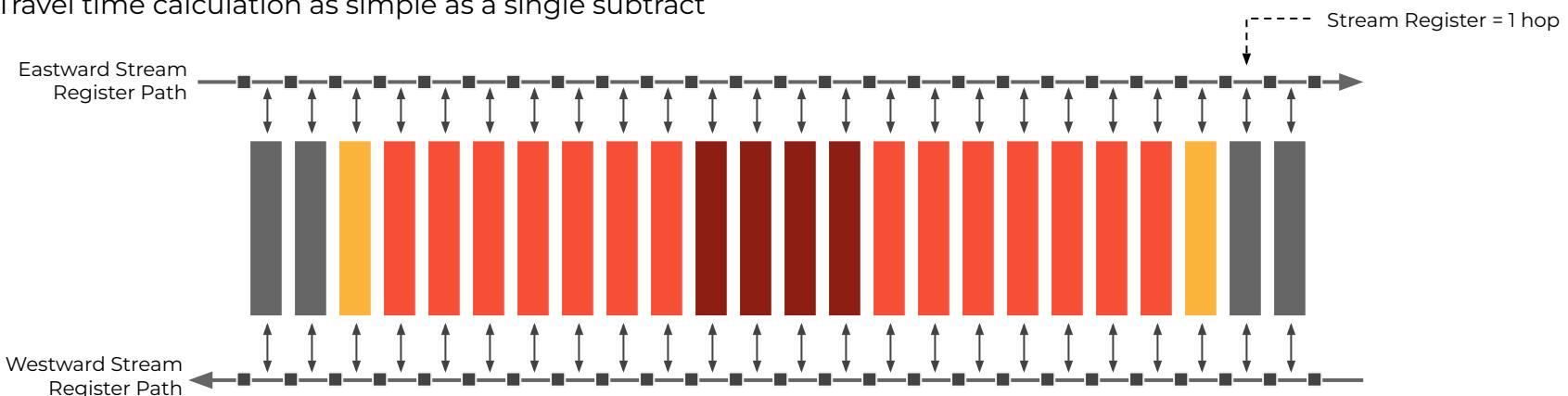
## **Compiler can quickly reason about all data movement between FUs**

Simple, one-dimensional interconnect for inter-FU communication

- Eastward and westward paths made up of arrays of “stream registers”
- Stream register = one-cycle hop

No arbiters/queues = software can easily reason about exact data movement without simulation

Travel time calculation as simple as a single subtract



# Data Type Support

Vector and Matrix compute units data types supported

Numerics for  
hardware-supported data types

Matrix (MXM) unit supports INT8  
and UINT8, and FLOAT16  
(half-precision)

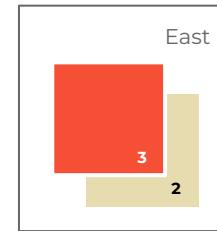
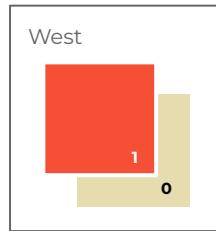
- 320-element **fused  
dot-product**
- INT32 or FP32  
accumulation

Vector (VXM) processor supports  
a superset of all data types

	Data Type	Lowest Value	Highest Value
	UNIT 8	0	255
	INT8	-128	127
	BOOL8	FALSE	TRUE
	UINT16	0	65,535
	INT16	-32,768	32,767
	BOOL16	FALSE	TRUE
	FLOAT16	-65,504	65,504
	UINT32	0	4,294,967,295
	INT32	-2,147,483,648	2,147,483,647
	BOOL32	FALSE	TRUE
	FLOAT32	-3.40E+38	3.40E+38

MXM

# Matrix Multiply Engines



Numeric Mode	Max Size	Supported Density	Result Tensor
int8	[N, 320] x [320, 320]	Two per MXM	int32
float16	[N, 320] x [160, 320]	One per MXM	float32

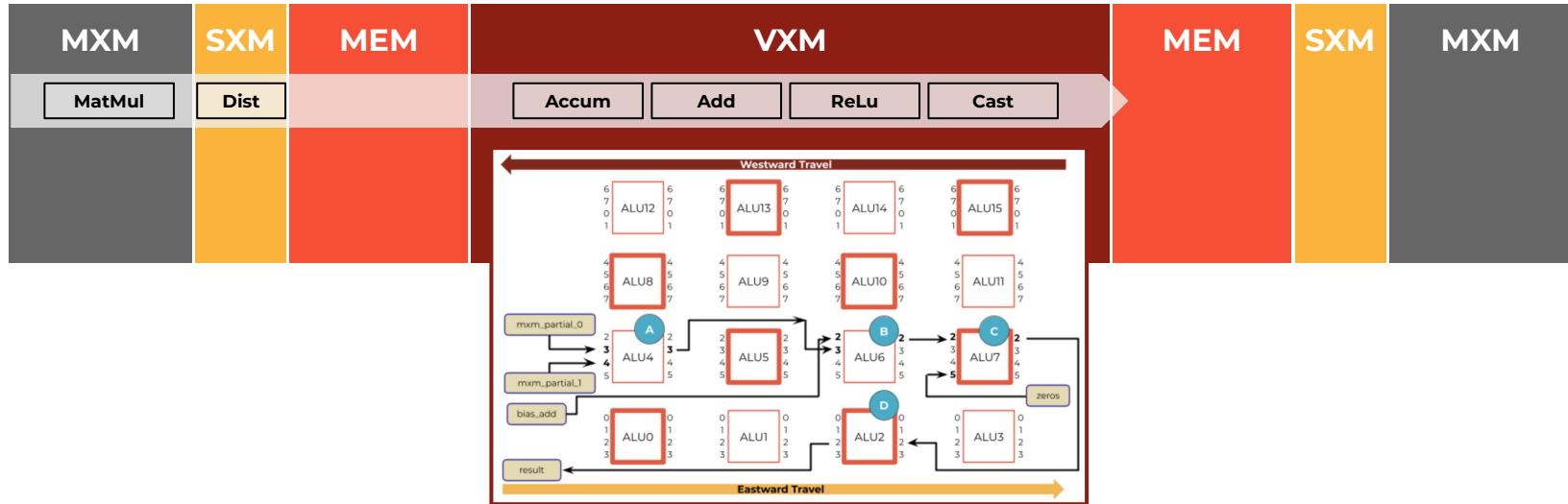
320B x 320B dot product  
Loads 320B x16 in 20 cycles  
20 cycle execution  
Fully pipelined, N

Int8 & float16  
Full precision expansion  
32-bit accumulate

Used Independently  
or together

VXM

# Vector Execution Module



Dataflow begins with memory  
Read onto Stream Tensor

Many concurrent streams  
are supported in  
programming model

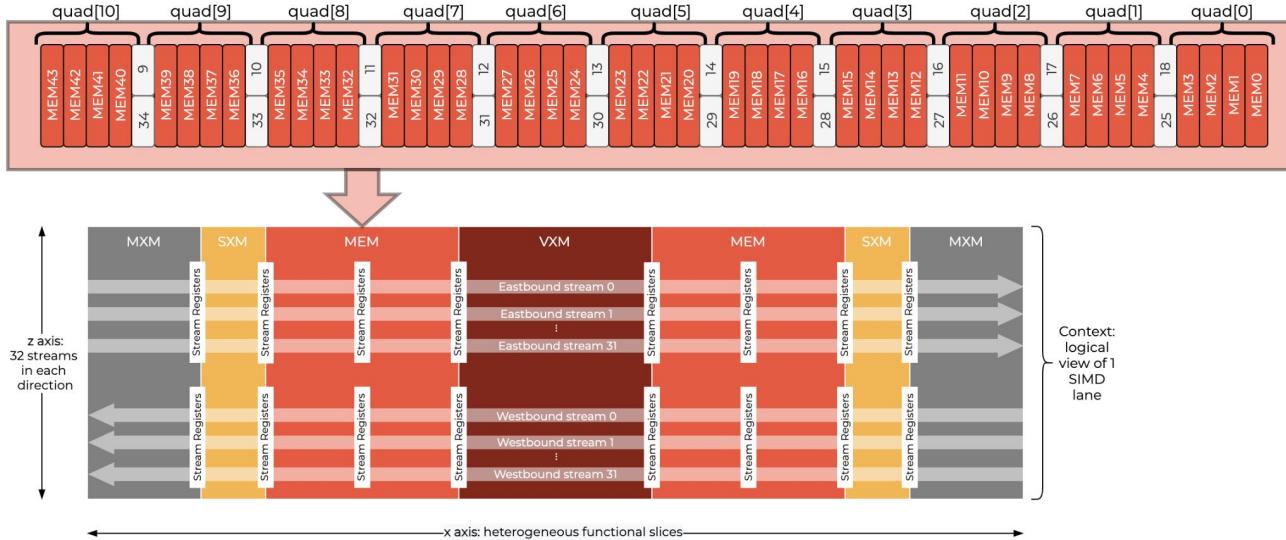
VXM provides a flexible  
and programmable fabric  
for Compute

Compute occurs on data  
locality of passing Stream Tensor

MEM bandwidth supports  
high concurrency

# MEM

# Memory



88 independent MEM slices with 8192 addresses (220MB) each arranged into quad timing groups

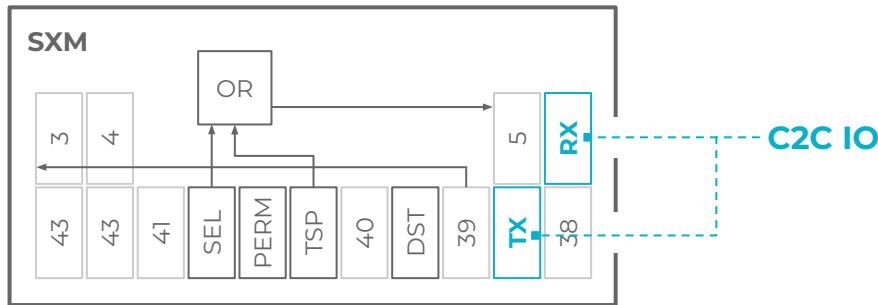
A read from a single MEM slice creates a 320Byte stream; a write terminates a stream

Group MEM slices for multi-dimensional tensors or multi-byte data types

Can read and write one physical stream (vector) per cycle, from 2 banks; Interfaces the full 64 stream bandwidth @ 80TBps

SXM

# Switch eXecution Module



Swiss army knife for data manipulation & Intra-vector byte operations

**Distributor:** 4 per hemisphere perform unto mapping of input + mask to output stream within a 16 byte superlane

**Transposer:** 2 per hemisphere perform intra-superlane transpose over 16 vectors for 20 superlanes

Permuter/Shifter: arbitrary mapping of input + mask, shuffling between 320B vector elements - used for data transforms like pads/reshapes

Shift, Rotate, Distribute, Permute, Transpose, Transport to SuperLanes

# Products:

## Seamlessly Scalable



GroqChip™ 1



GroqCard™



GroqNode™



GroqRack™

# Designing for Determinism

Building hardware to be an efficient compiler target

## Design choices:

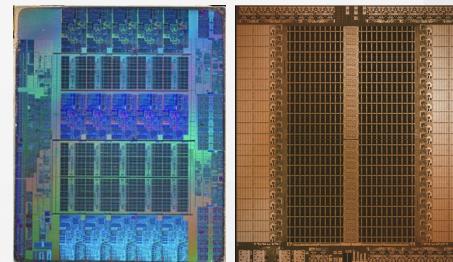
- **Explicitly removing “spins & waits,”** or unpredictable execution, from the architecture
- **Reactive components** e.g. speculative execution, memory cache, etc

Hardware then enables the compiler and runtime interfaces to **reason about program execution**

- Enables the Compiler to ‘know’ the exact state of every tensor on-chip
- Allows Compiler to efficiently orchestrate the operands and instructions
- Enables programmers dependably extract performance
- Delivers predictable execution model at scale (100s, 1000s of devices)

### Conventional CPUs Add Features and Complexity

Speculative execution;  
cached memory



### GroqChip™ Simplifies Data Flow Through Stream Programming

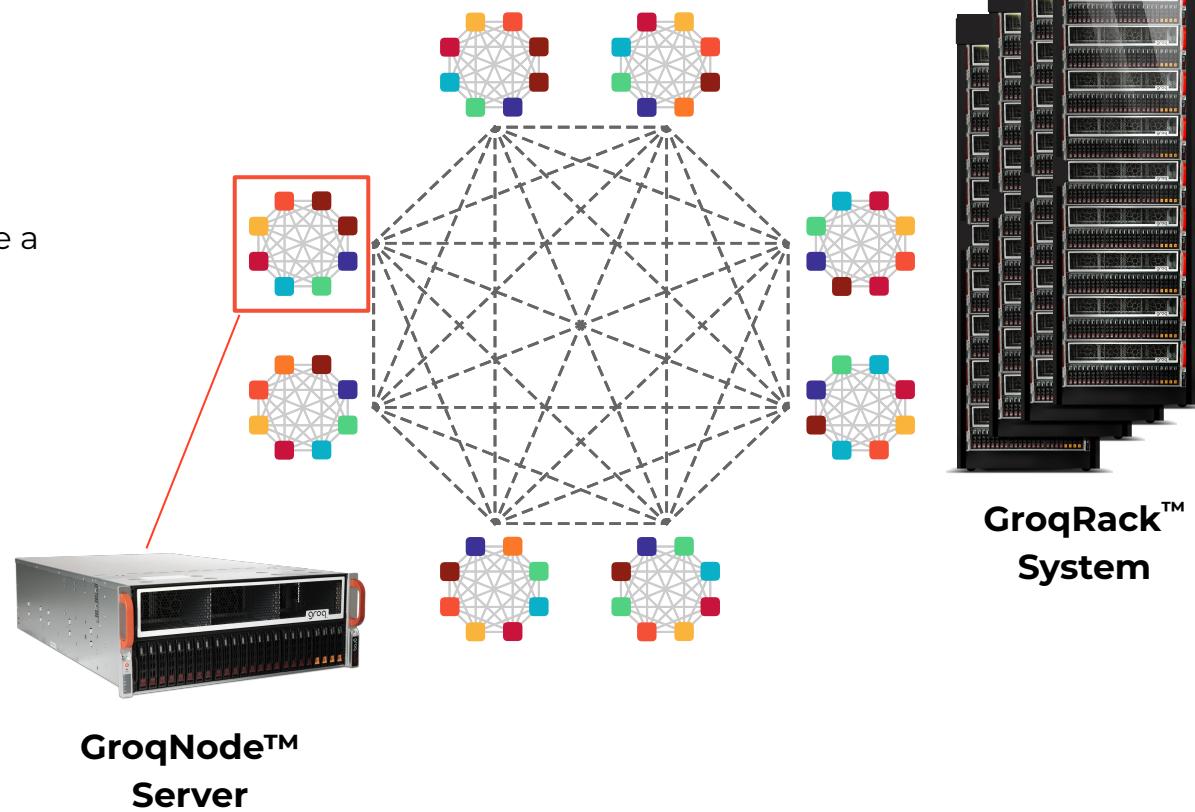
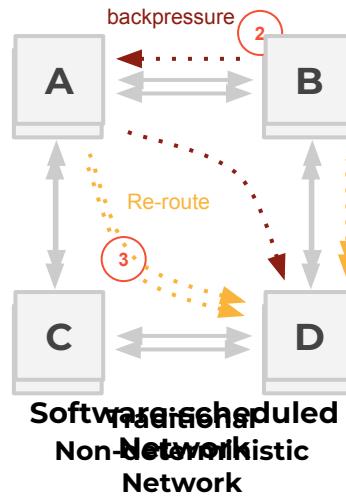
Predictable performance  
at scale

# Groq Scale-out

## Building GroqRack™

Collection of nodes are used to create a “group” or a **virtual** high-radix router

No network congestion or adaptive routing



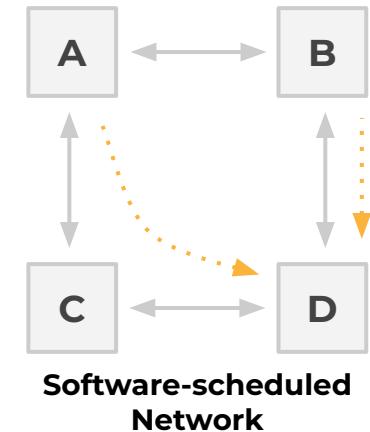
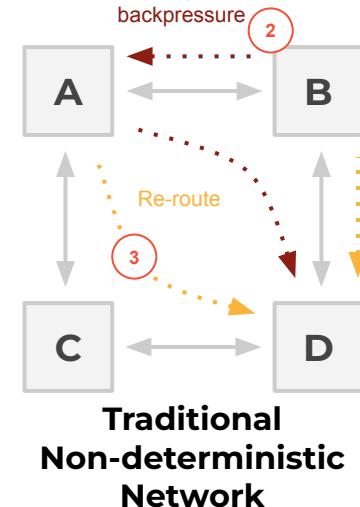
# Deterministic Adaptive Routing

## Conventional Network

- Commonly done based on network backpressure
- Reactive approach makes the routing decision difficult, increases latency, and increases hardware complexity
- Network latency is **unpredictable**

## Software-scheduled Network

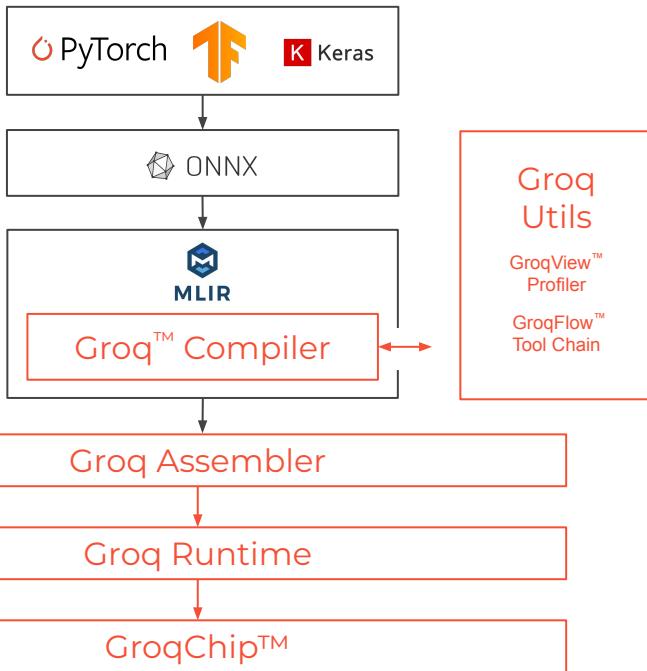
- Avoids congestion
- Enables maintaining a deterministic Tensor Streaming Processor architecture to scale to a multi-node deterministic network execution



# GroqWare™ Suite Intro

# GroqWare™ Suite

GroqFlow™



DIVERSE SUITE OF  
DEVELOPMENT TOOLS

## Out-of-Box

**Groq Compiler** provides out-of-the-box support for standard Deep Learning models



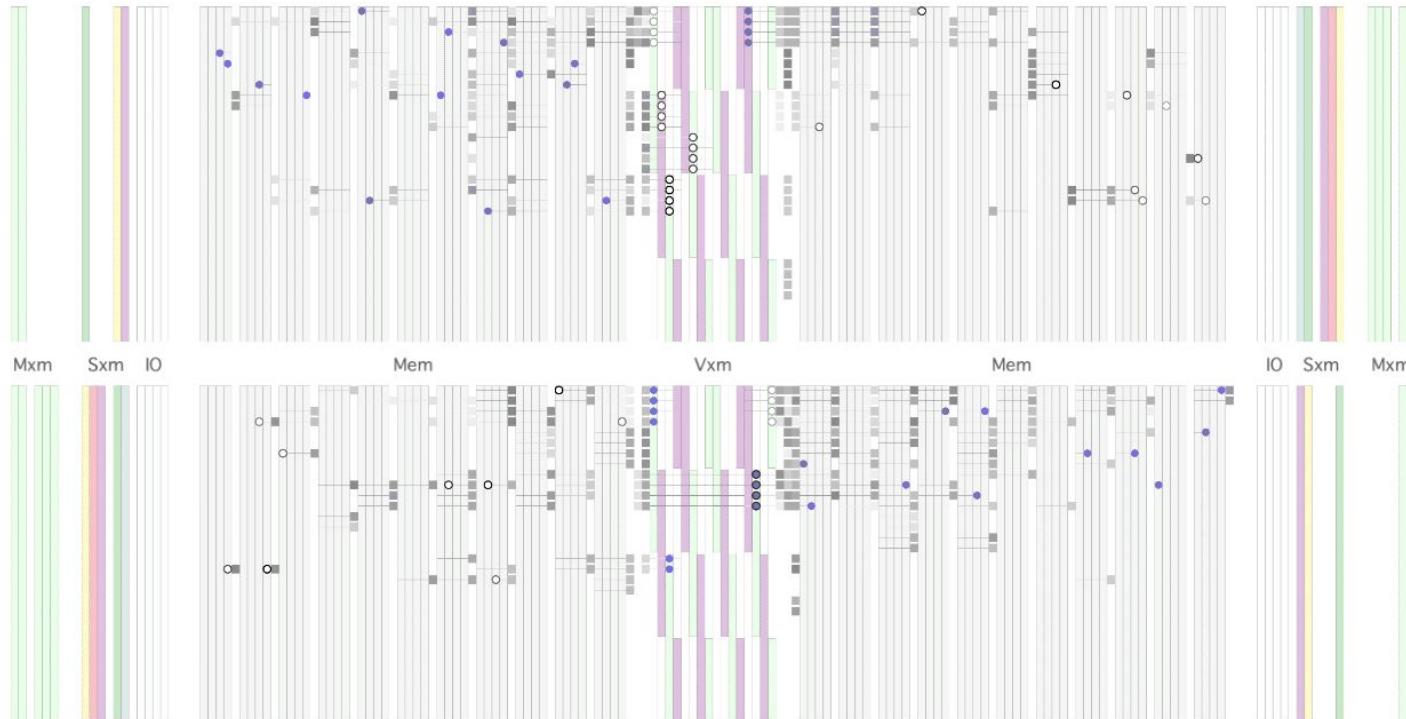
## Productivity Tools

**GroqView Profiler** provides visualization of the chip's compute and memory usage at compile time

**GroqFlow Tool Chain** enables a single line of Pytorch or TensorFlow code to import and transform models through a fully automated tool chain to run on Groq hardware

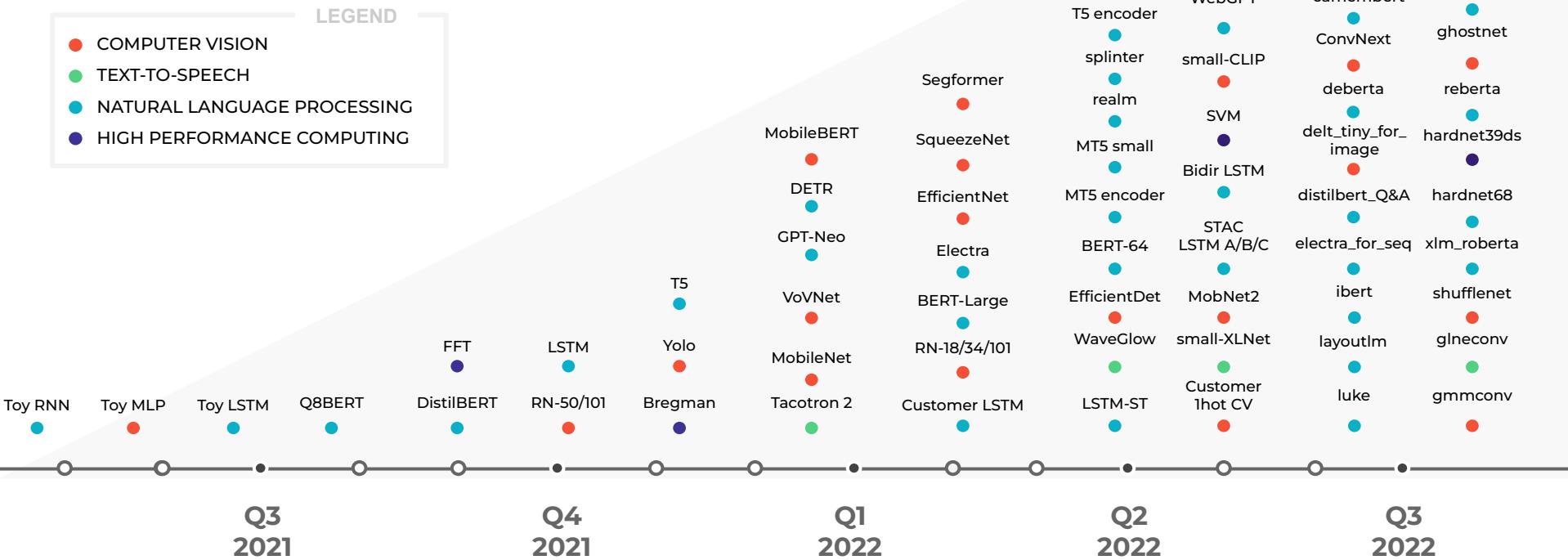
# Visualizing Data Orchestration

Given to Groq Compiler



# Exponentially Supporting More Workloads

What we've enabled with Groq™ Compiler thus far\*



# Groq Systems

Unlocking new possibilities at the intersection of HPC and ML workloads

Customer Workload	Value delivered
<b>Drug Discovery:</b> Co-resident / ensemble models	<b>Up to 200x speed up</b> when evaluating candidate COVID drugs <sup>(1)</sup>
<b>Cyber Security:</b> Co-resident / ensemble models	<b>Up to 2,000x speed up</b> for real-time threat detection <sup>(2)</sup>
<b>Fusion Energy:</b> LSTM for controls systems	<b>Up to 600x speed up</b> to make real-time plasma stabilization possible <sup>(3)</sup>
<b>US Government:</b> High resolution FFTs	<b>Up to Real time, 750ms</b> , calculation of 385M point FFTs
<b>Capital Markets:</b> X <sup>T</sup> X - core of regression analysis	<b>Up to 100x faster iteration</b> on 10-100TB of data to sharpen predictions



Hugging Face  
Natural Language Processing



ONNX Model Zoo  
Computer Vision



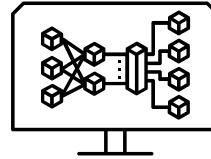
PyTorch Geometric  
Graph Neural Networks



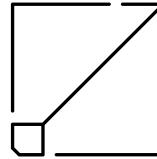
Hummingbird  
Classical ML

## WHERE IS GROQ GOING?

Our  
Vision



ANY MODEL



ANY SIZE



PEAK  
PERFORMANCE

EASE OF USE

# GroqFlow™

PyPi package: <https://pypi.org/project/groqflow/4.2.0/> `pip install groqflow`  
GitHub repository: <https://github.com/groq/groqflow>

# 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

# Introducing GroqFlow™

## Groqlt Key Functions

**COMPILES** models  
into Groq programs

gmodel = groqlt(model,inputs)



**EXECUTES** programs  
on GroqChip™

gmodel(\*\*inputs)

**BENCHMARKS** programs  
on GroqChip (Coming soon)

latency = gmodel.benchmark()

# Introducing GroqFlow™

## Groqlt Key Functions

**COMPILES** models  
into Groq programs

```
gmodel = groqlt(model,inputs, num_chips=2)
```

**EXECUTES** programs  
on GroqChip™

```
gmodel(**inputs)  
(No change!)
```

**BENCHMARKS** programs  
on GroqChip (Coming soon)

```
latency = gmodel.benchmark()  
(No change!)
```



Search or jump to...

Pull requests Issues Marketplace Explore



groq/groqflow Public

Watch 1

Fork 0

Starred 8

Code Issues Pull requests Discussions Actions Wiki Security Insights

main

1 branch

0 tags

Go to file

Add file

Code

 vgodsoe-groq	Update readme.md	75beaaf 7 days ago	2 commits
 docs	GroqFlow Release 2.1.1	8 days ago	
 examples	GroqFlow Release 2.1.1	8 days ago	
 groqflow	GroqFlow Release 2.1.1	8 days ago	
 license.md	GroqFlow Release 2.1.1	8 days ago	
 readme.md	Update readme.md	7 days ago	
 setup.py	GroqFlow Release 2.1.1	8 days ago	

readme.md

## GroqFlow

GroqFlow™ is the easiest way to get started with Groq's technology. GroqFlow provides an automated tool flow for compiling machine learning and linear algebra workloads into Groq programs and executing those programs on GroqChip™ processors.

We recommend that your system meets the following hardware requirements:

- To build models: 32GB or more of RAM.
- To run models: at least 1 GroqChip processor. For larger models, additional GroqChip processors may be required (1, 2, 4, and 8).

### About

GroqFlow provides an automated tool flow for compiling machine learning and linear algebra workloads into Groq programs and executing those programs on GroqChip™ processors.

 Readme View license 8 stars 1 watching 0 forks

### Releases

No releases published

### Packages

No packages published

### Languages

 Python 100.0%

 [groq / groqflow](#) Public

Groq Inc. groq

9 4 repositories 1 member

Discussions Actions Security Insights

0 tags Go to file Code

vgodsoe-groq GroqFlow Release 2.1.1 842415e 1 hour ago 1 commit

docs	GroqFlow Release 2.1.1	1 hour ago
examples	GroqFlow Release 2.1.1	1 hour ago
groqflow	GroqFlow Release 2.1.1	1 hour ago
license.md	GroqFlow Release 2.1.1	1 hour ago
readme.md	GroqFlow Release 2.1.1	1 hour ago
setup.py	GroqFlow Release 2.1.1	1 hour ago

readme.md

## GroqFlow 🚀

GroqFlow™ is the easiest way to get started with Groq's technology. GroqFlow provides an automated tool flow for compiling machine learning and linear algebra workloads into Groq programs and executing those programs on GroqChip™ processors.

We recommend that your system meets the following hardware requirements:

- To build models: 32GB or more of RAM.
- To run models: 8 GroqChip processors is recommended, especially for larger models.

## Installation Guide

About

No description, website, or topics provided.

Readme View license 1 star 0 watching 0 forks

Releases

No releases published

Packages

No packages published

Languages

Python 100.0%



PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

bash - groqflow + × ☰ ^ ×

○ (my\_venv) ubuntu@groqit-cloud:~/awesome/groqflow\$ █



# Models Running on Groq™ Compiler

Out-of-the-box support

## Graph Convolutions (PyTorch Geometric):

- GINEconv
- GMMConv
- PNACConv
- SAGEConv
- GINConv
- CGConv

## Hugging Face Transformers:

- BERT
- CamemBERT
- ConvNeXt
- DeBERTa
- DistilBERT
- Electra
- GPT-1/2
- I-BERT
- LayoutLM
- LUKE
- MiniLMv2
- MobileBERT
- REALM
- RoBERTa
- SegFormer
- Speech2Text
- Splinter
- T5 Encoder
- ViT Base
- XLM-RoBERTa

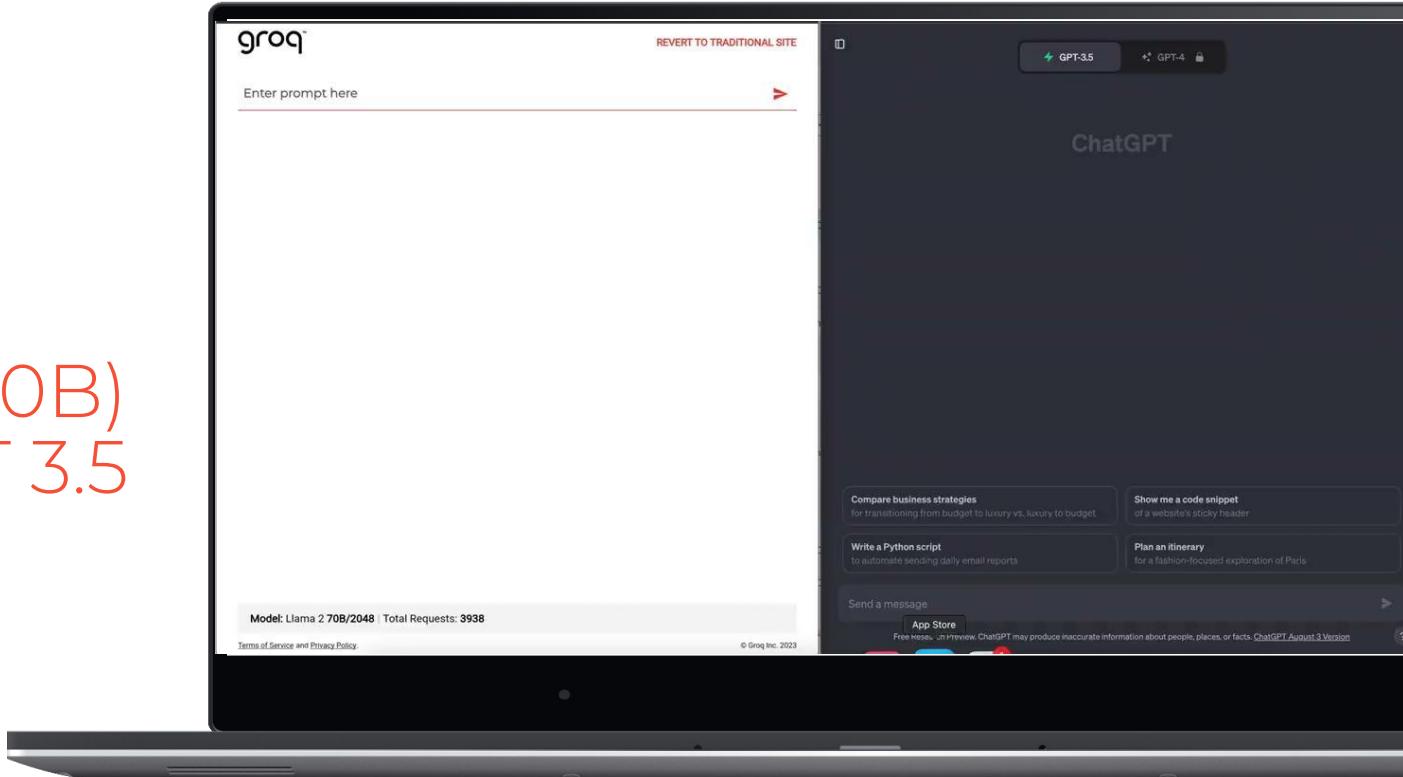
## Others (Vision Models):

- GhostNet
- GoogLeNet
- HarDNet 39-DS/68/68-DS
- LeViT 128/192/256/384
- MEAL V1/V2
- MobileNet v2
- ProxylessNAS CPU/GPU/Mobile
- ResNet 18/34/50/101
- Shufflenet V2 x1.0
- SqueezeNet 1.0/1.1

# LLMs on Groq

## Head-to-Head

GroqChat  
(LLama-2 70B)  
vs ChatGPT 3.5  
Turbo



# Groq API



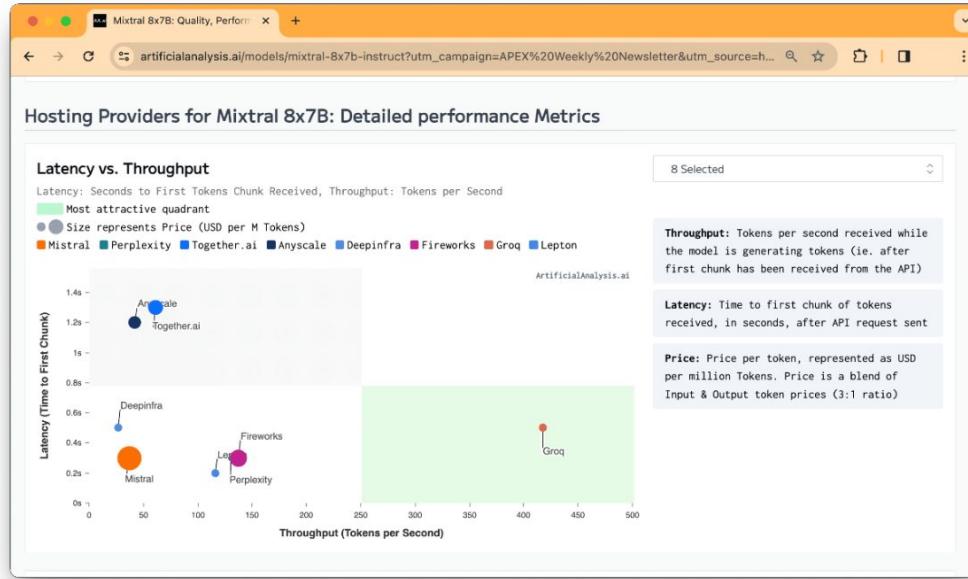
Hatice Ozen, Sr. Customer Apps Engineer



# 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.



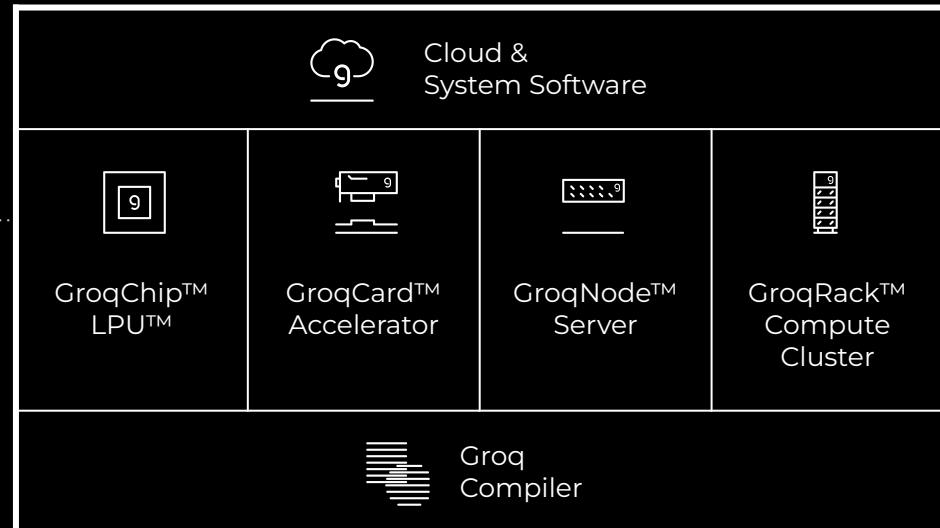
≡ AI Inference Solution

# Vertically Integrated AI Inference Solution.



## AI Inference Co-cloud

# groqcloud™



groq®

~~223K~~ 295K

USERS >

Now developing  
on GroqCloud.



## ☰ How to get access to GroqChat?

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

☰ How to get access to GroqChat?

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

Groq is Fast AI Inference

groq

Try it

What is AI inference

What is generative AI

Why fast inference matters

Chat with us

Sign in to Groq

LPUTM AI INFERENCE ENGINE

LLM-based chatbot. Accuracy, correctness or appropriateness cannot be guaranteed.

Why Groq

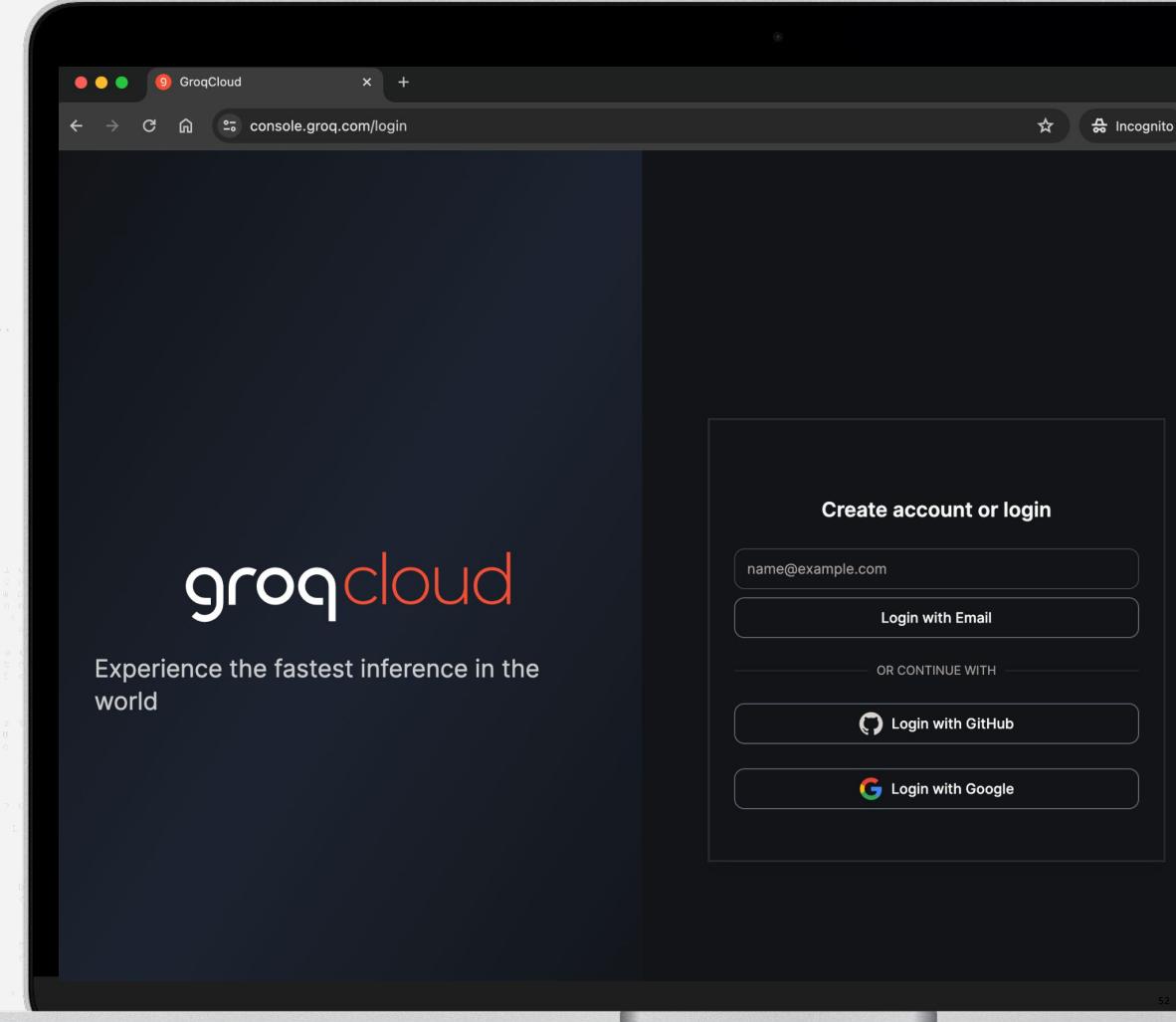
GroqCloud

More

☰ 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 circular diagram representing a learning tree. The root node is "Photography Basics", which branches into "Equipment", "Composition", and "Technique". "Equipment" further branches into "Cameras", "Lenses", and "Accessories". "Composition" branches into "Rule of Thirds", "Depth of Field", and "Framing". "Technique" branches into "Exposure", "Color Theory", and "Lighting". A callout box highlights the "Exposure" node, showing its sub-components: "Aperture", "Shutter Speed", and "ISO". Below the diagram, a text post by Felipe Schieber reads:

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



## DEMO: Structuring Generated Content

A screenshot of a video player interface. The video content is a presentation slide titled "CMOS Technology: A Brief History and Overview". The slide text discusses the history of CMOS technology, starting with the design phase and mentioning the transition from discrete components to integrated circuits. It highlights the invention of the metal-oxide-semiconductor field-effect transistor (MOSFET) in 1963 by Frank Wanlass, which allowed for the creation of complex logic gates. The slide also notes that CMOS technology has evolved to offer higher performance and lower power consumption compared to earlier technologies like TTL. The video player interface includes a play button, a progress bar showing 0:21 / 1:07, volume controls, and other standard video player controls.

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-cookbo  
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



Discord

discord.com/invite/groq

9

You've been invited to join

**GroqCloud**

2,440 Online • 16,832 Members

DISPLAY NAME

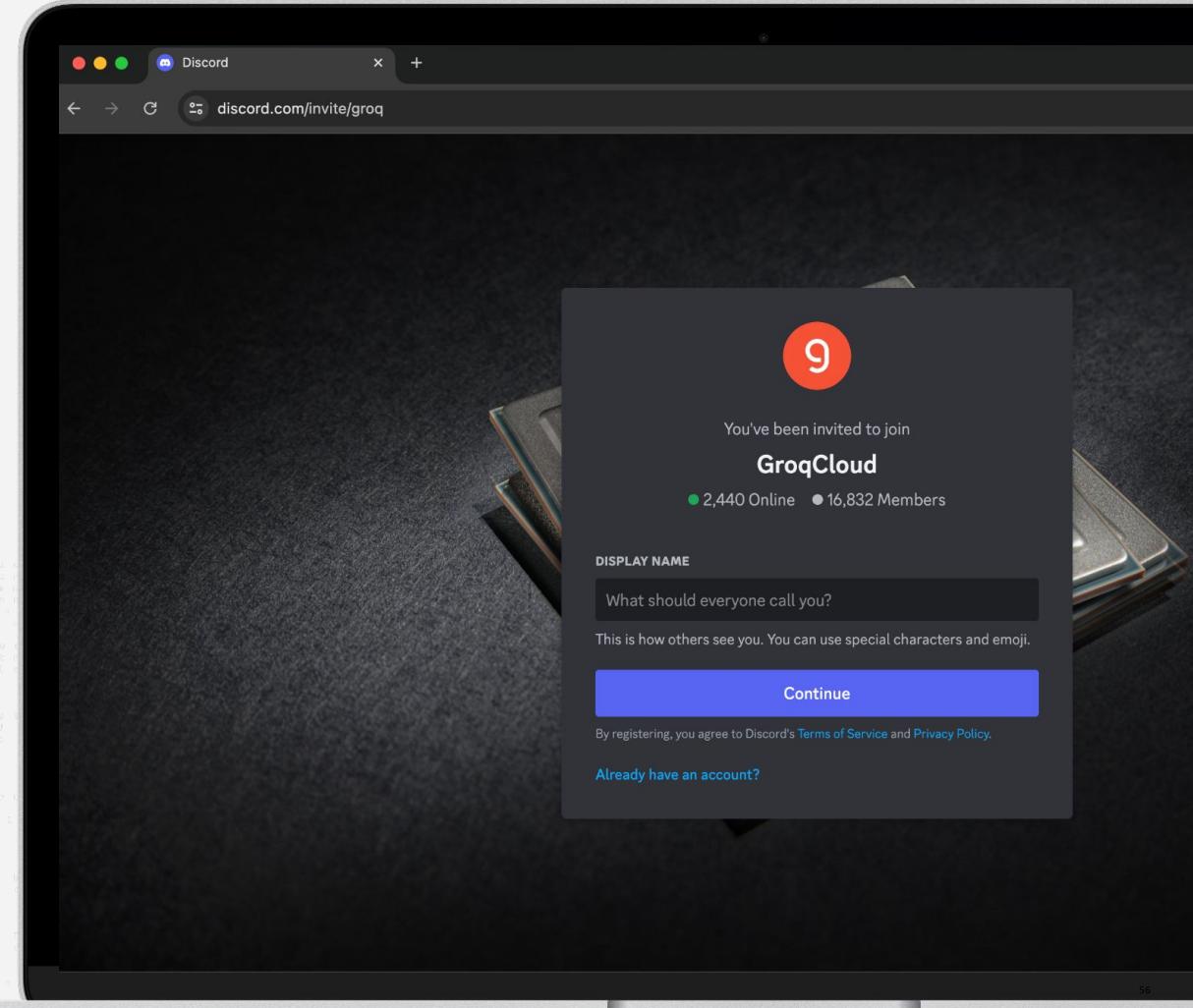
What should everyone call you?

This is how others see you. You can use special characters and emoji.

Continue

By registering, you agree to Discord's [Terms of Service](#) and [Privacy Policy](#).

Already have an account?



## 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 Inc. 2024



The Groq logo is centered on a light gray background. It consists of the word "groq" in a bold, lowercase, sans-serif font. A small registered trademark symbol (®) is positioned at the top right of the letter "q". The background features a subtle pattern of overlapping gray squares of varying sizes, creating a pixelated or grid-like texture.

groq®