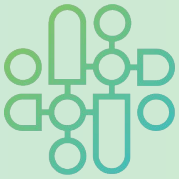


Programming Novel AI Accelerators for Scientific Computing

Tutorial at Supercomputing 2023
11 November 2023 Denver, CO USA

Sanjif Shanmugavelu (sshanmugavelu@groq.com), Software Engineer (Maxeler)

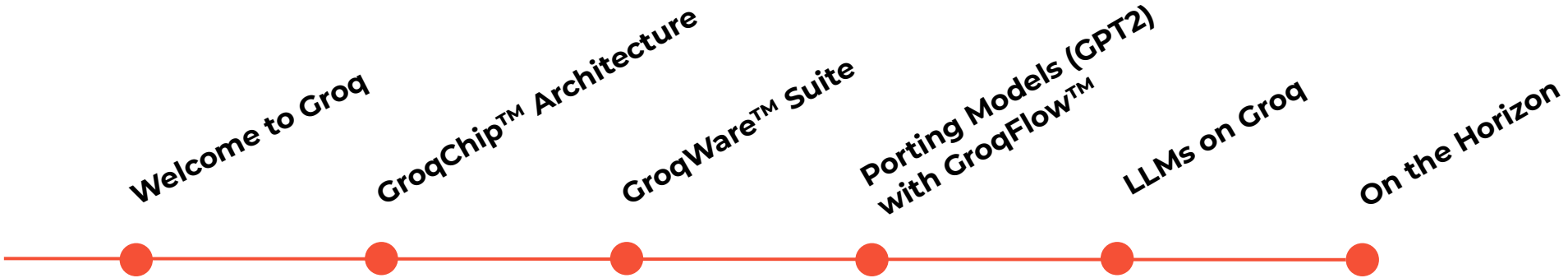
groq™

Programming Novel AI Accelerators for Scientific Computing

Tutorial at Supercomputing 2023
11 November 2023 Denver, CO USA

Sanjif Shanmugavelu (sshanmugavelu@groq.com), Software Engineer (Maxeler)

Agenda



Who is Groq?

WE ARE

>>> an AI-born

>>> software-first platform

>>> made to run real-time

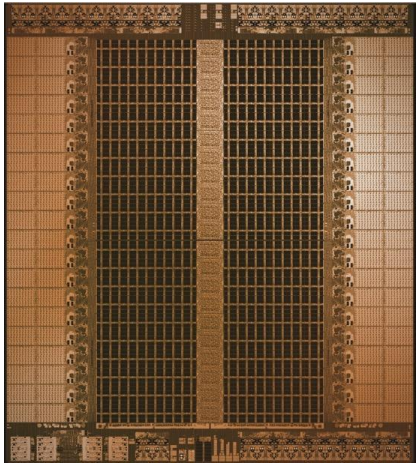
>>> AI solutions at scale

groq™

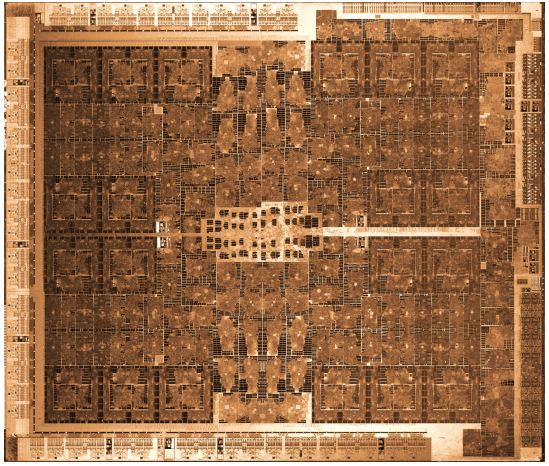
GroqChip™ Architecture

Simply Better

Language Processing Unit™ Accelerator GroqChip™ 1

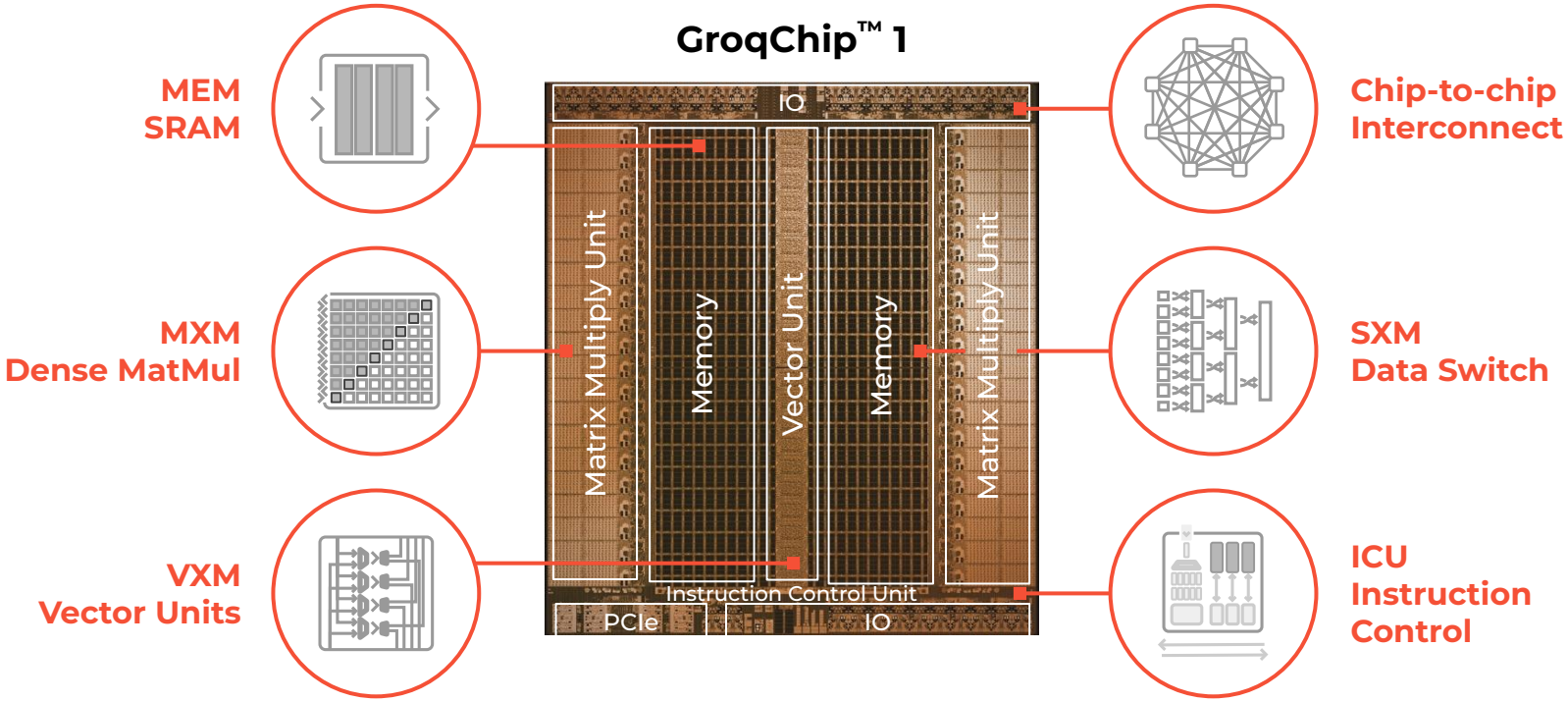


Typical Graphics Processor (GPU)



GroqChip™ At-a-glance

Scalable compute architecture



GroqChip™ 1 Overview

Scalable compute architecture

SRAM Memory

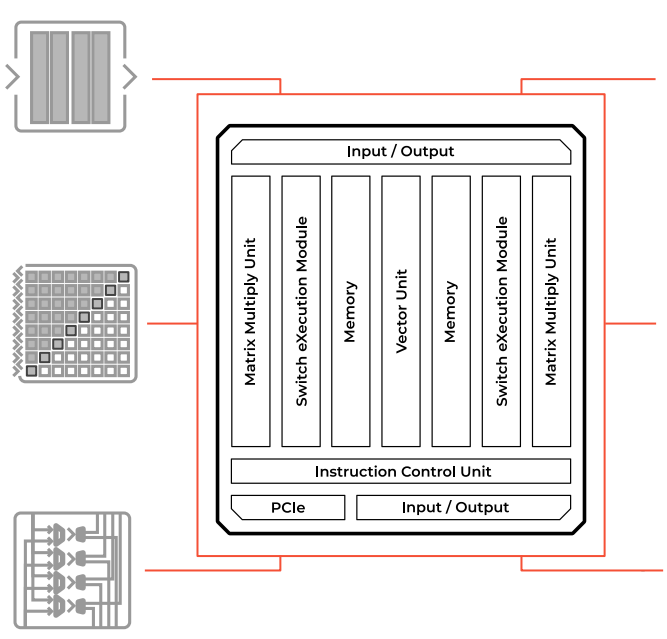
Massive concurrency
80 TB/s of BW
230MB capacity
Stride insensitive

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

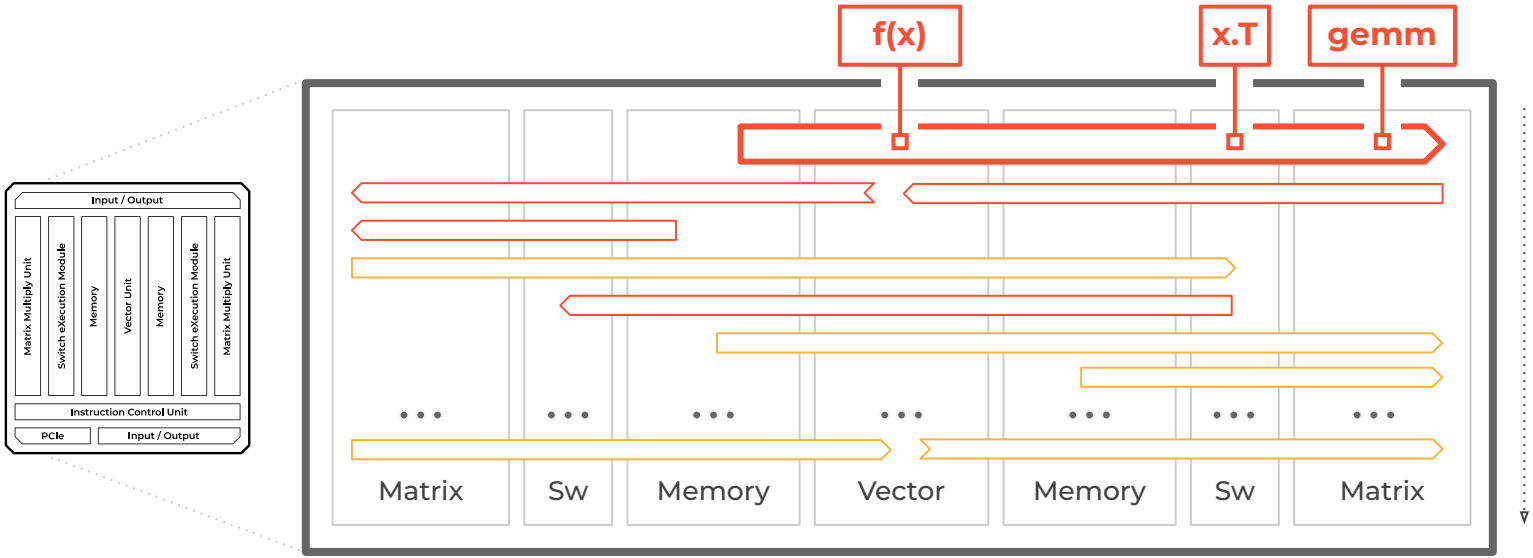
Shift, Transpose, Permuter for improved data movement and data reshapes

Instruction Control

Multiple instruction queues for instruction parallelism

Radically Simplifying Compute

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



LOW LATENCY

PREDICTABLE PERFORMANCE

SCALABLE COMPUTE

Single core, spatial pipeline processing fabric

Simple tensor instruction set arch

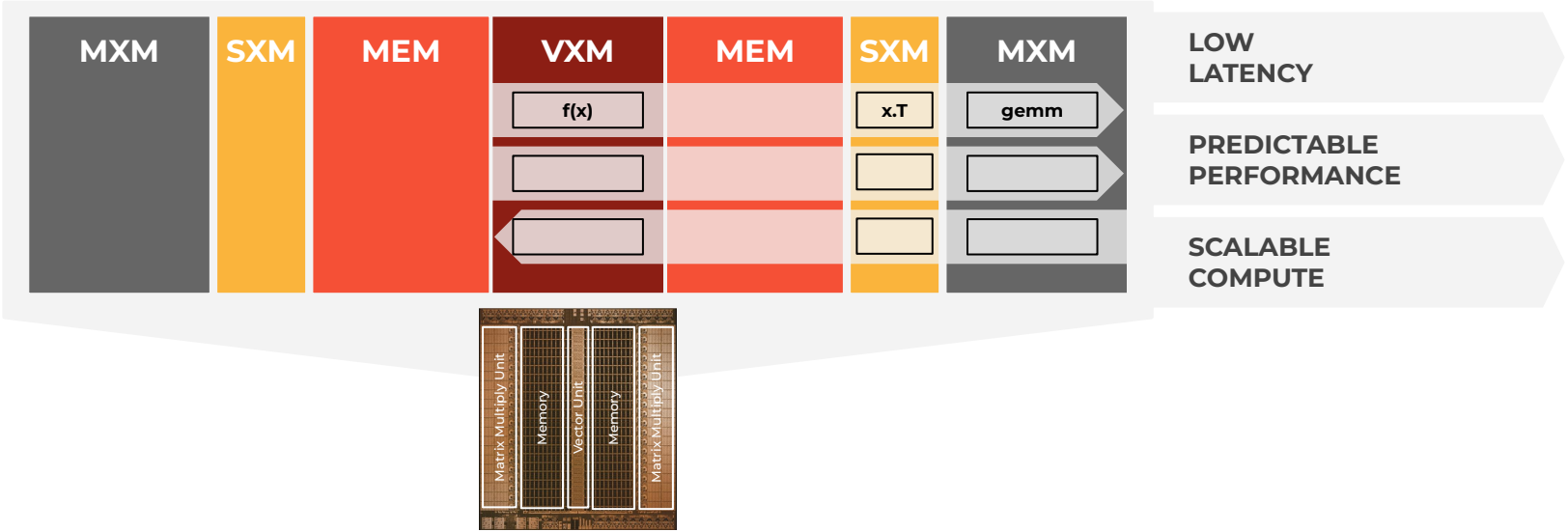
Large on-chip memory bandwidth

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 * fclk
= 1024 TOps

(1 Peta Op per Second @1.25GHz)

Memory

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

44*2*4096*2*320 = 220MB

Memory-Compute Units BW

BW = Number of streams in parallel in both directions * Bytes/stream * Clock * hemispheres * Computation units

= 64 * 320 *900 MHz * 2 * 2 = 80TB/s

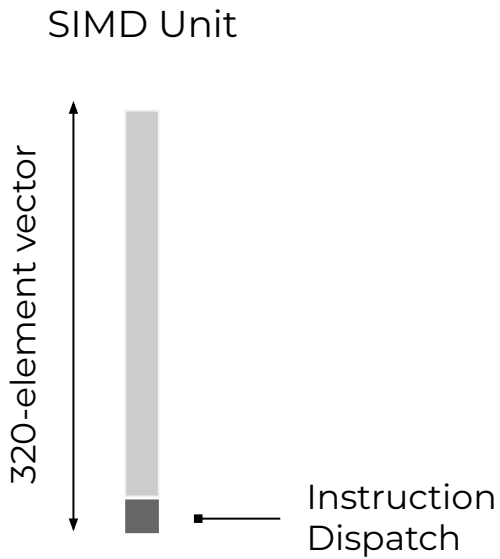
IO BW

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

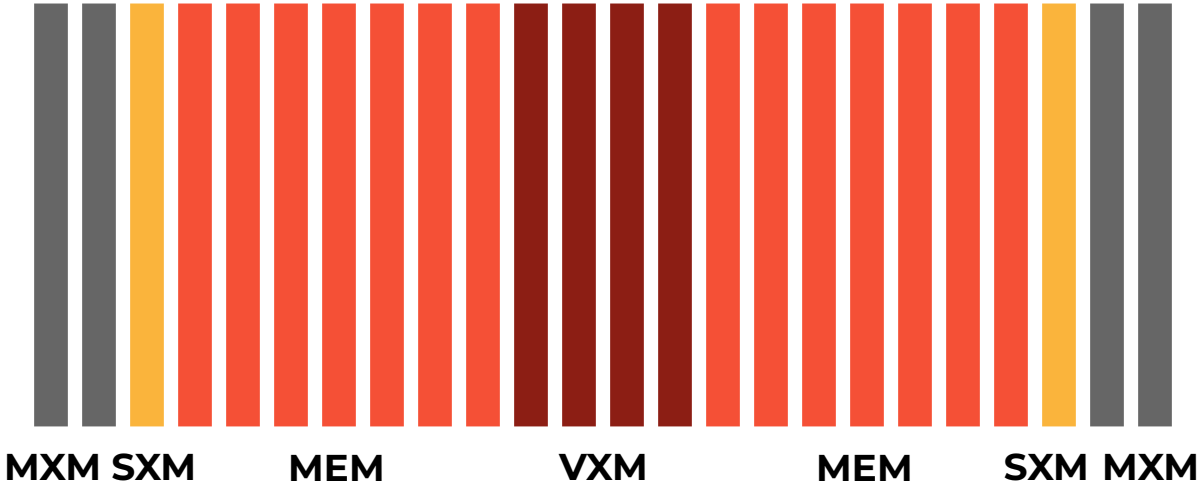
VXM
Vector-Vector
Operations

SXM
Data Reshapes

MEM
On-chip SRAM

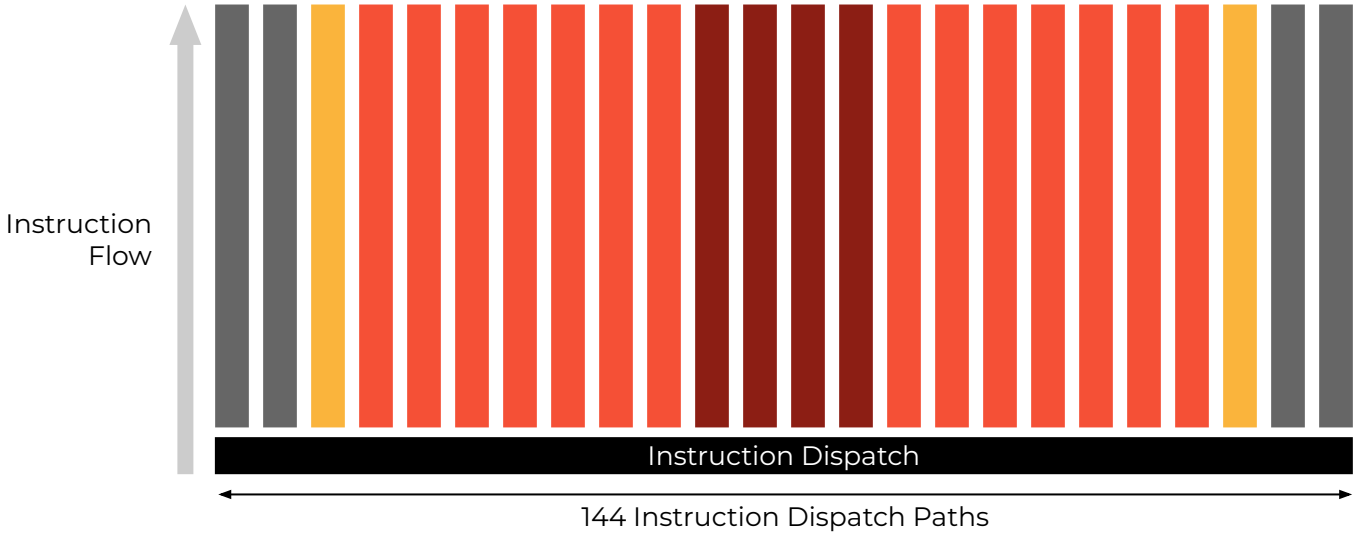
GroqChip™ Building Blocks

Lay out SIMD units across chip area



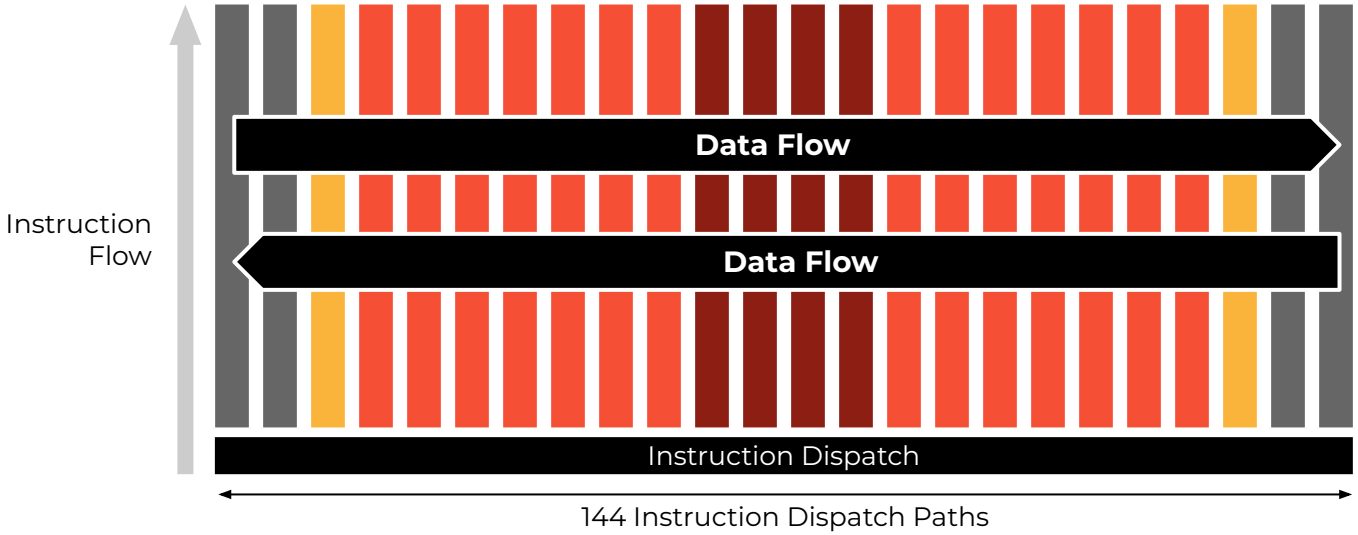
GroqChip™ Building Blocks

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

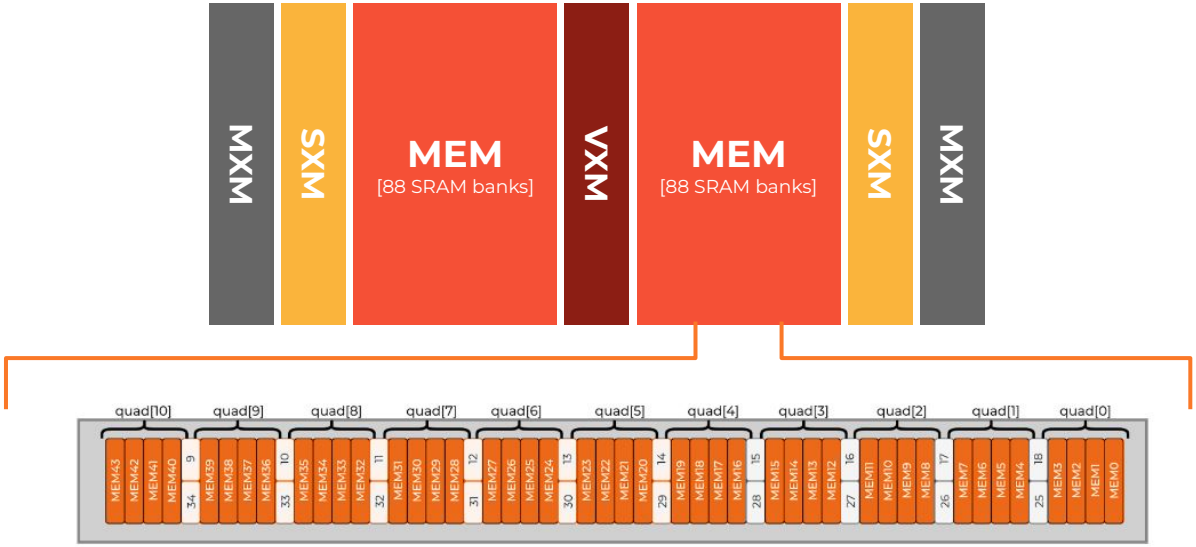
- Compiler aware of all data locations at any given point in time

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

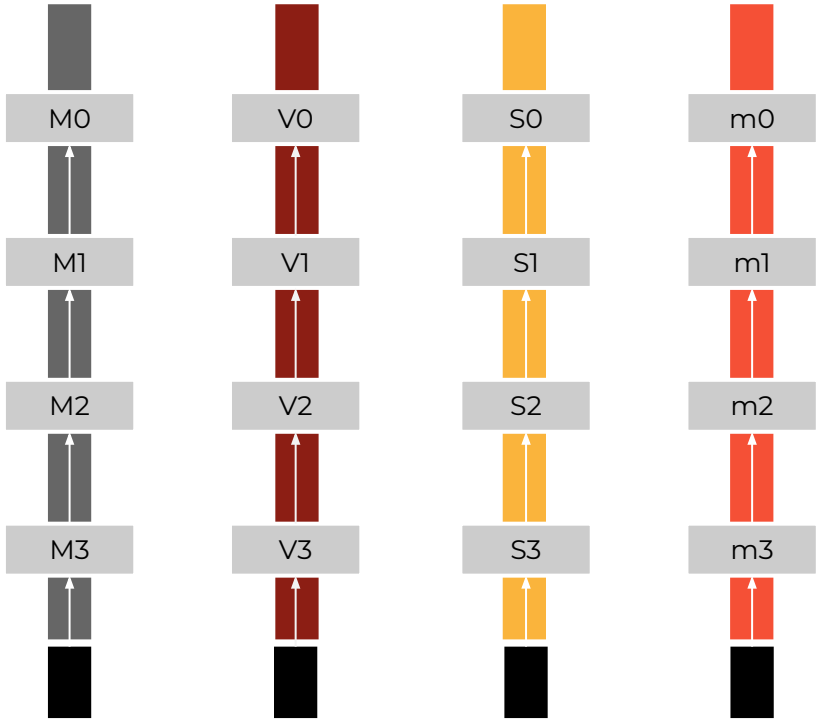
Compiler empowered to perform Cycle-accurate instruction scheduling

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

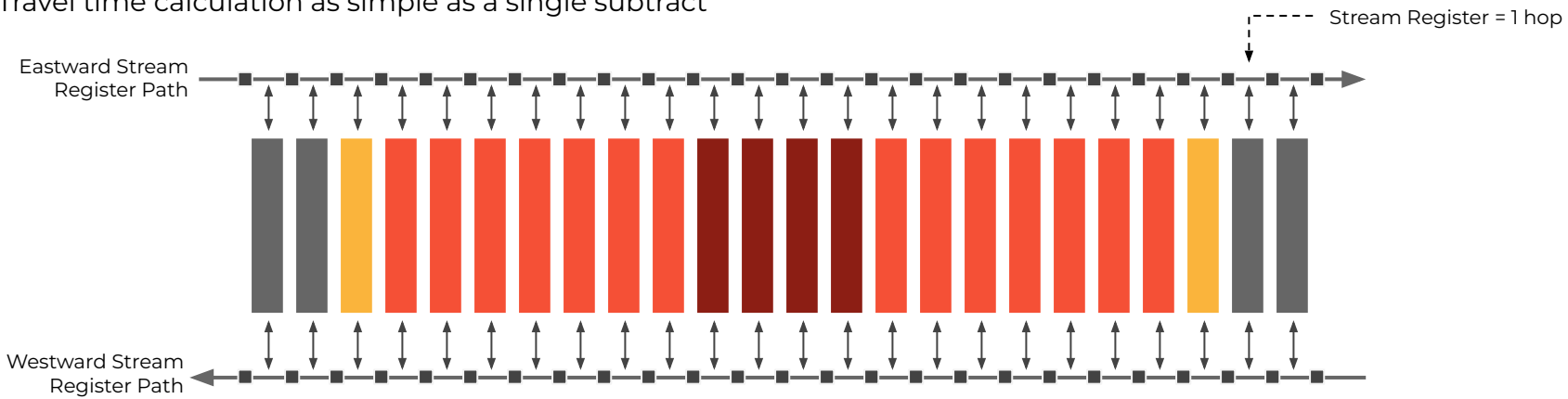
Compiler can quickly reason about all data movement between FUs

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)

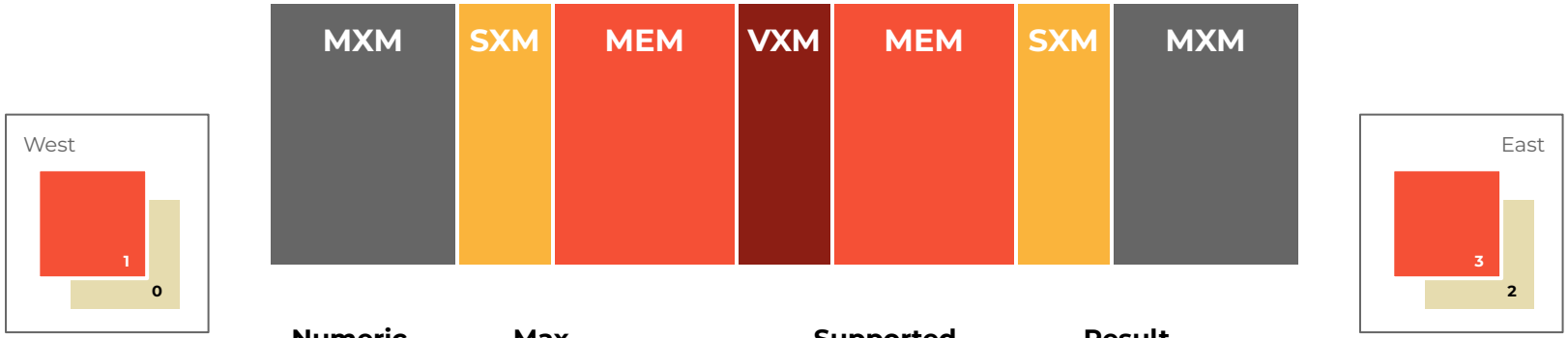
- 320-element **fused dot-product**
- INT32 or FP32 accumulation

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

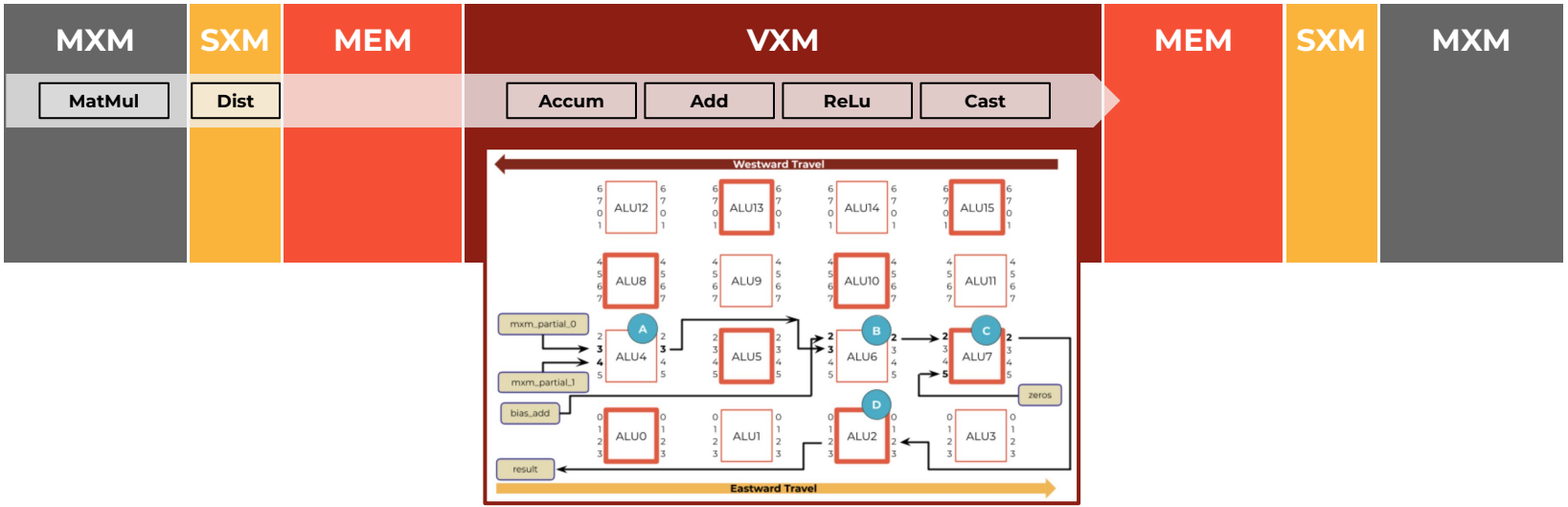
320B x 320B dot product
Loads 320B x16 in 20 cycles
20 cycle execution
Fully pipelined, N

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

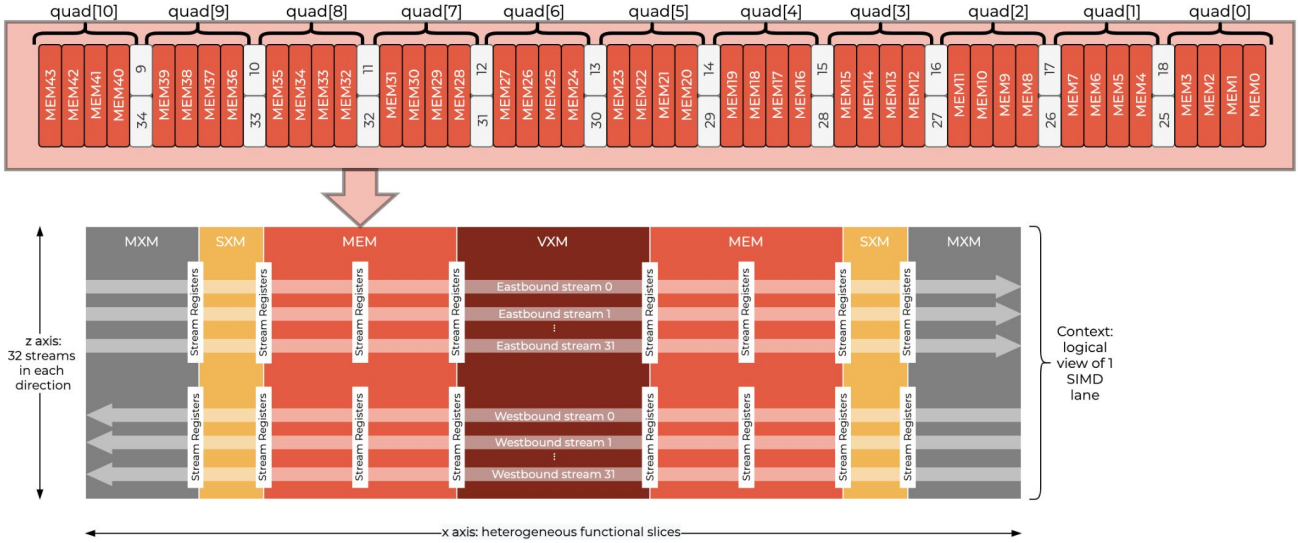
VXM provides a flexible and programmable fabric for Compute

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

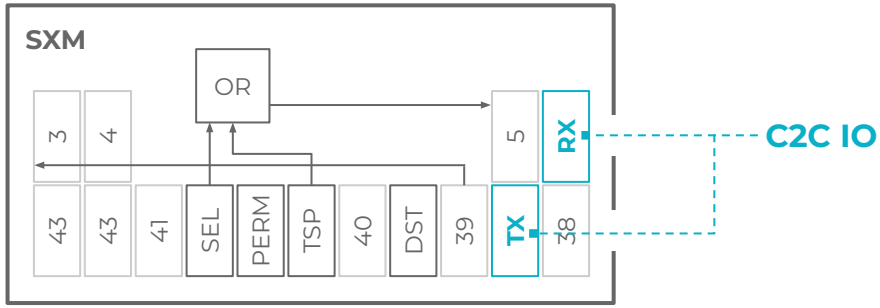
A read from a single MEM slice creates a 320Byte stream; a write terminates a stream

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:

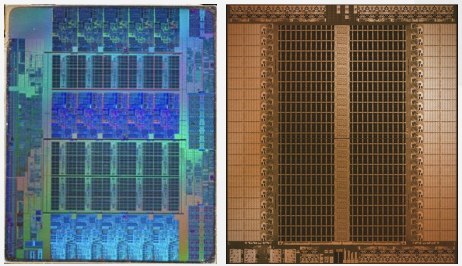
- **Explicitly removing "spins & waits,"** or unpredictable execution, from the architecture
- **Reactive components** e.g. speculative execution, memory cache, etc

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

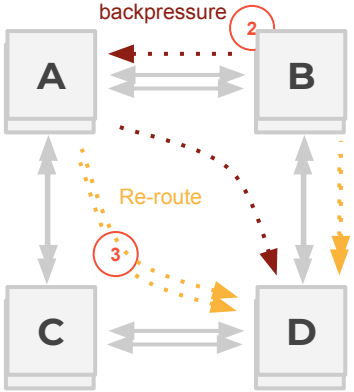
Predictable performance at scale

Groq Scale-out

Building GroqRack™

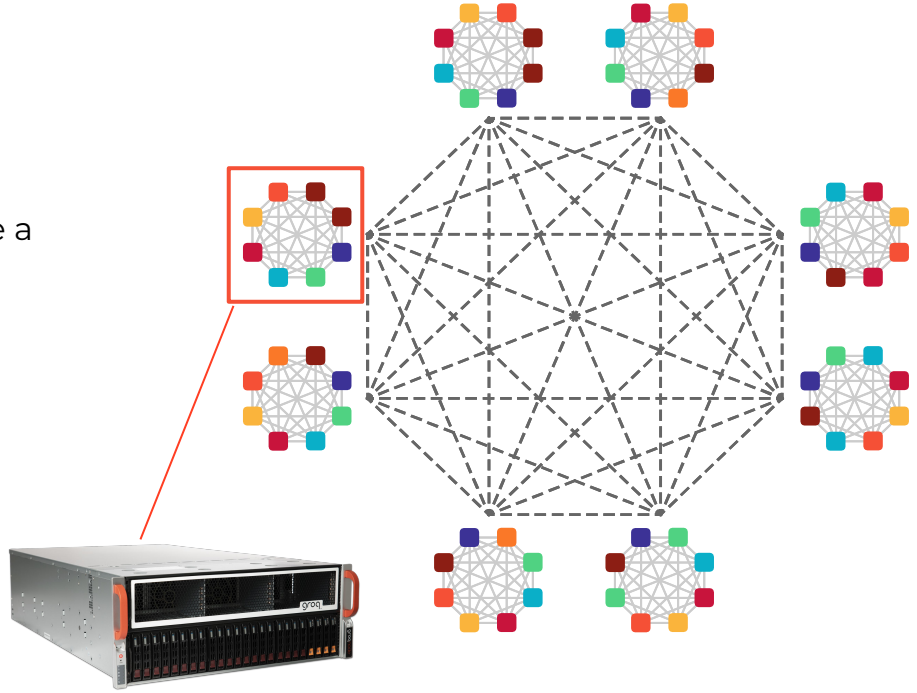
Collection of nodes are used to create a “group” or a **virtual** high-radix router

No network congestion or adaptive routing

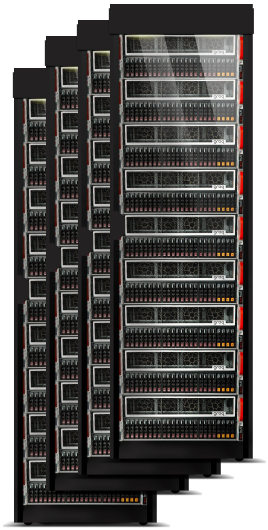


Software-scheduled Non-deterministic Network

Traditional Network



GroqNode™ Server



GroqRack™ System

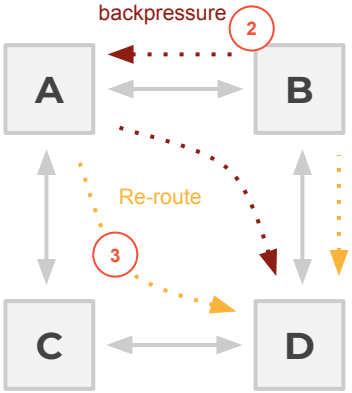
Deterministic Adaptive Routing

Conventional Network

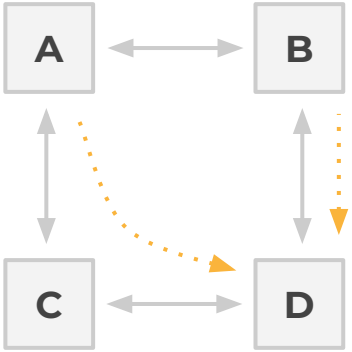
- Commonly done based on network backpressure
- Reactive approach makes the routing decision difficult, increases latency, and increases hardware complexity
- Network latency is **unpredictable**

Software-scheduled Network

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



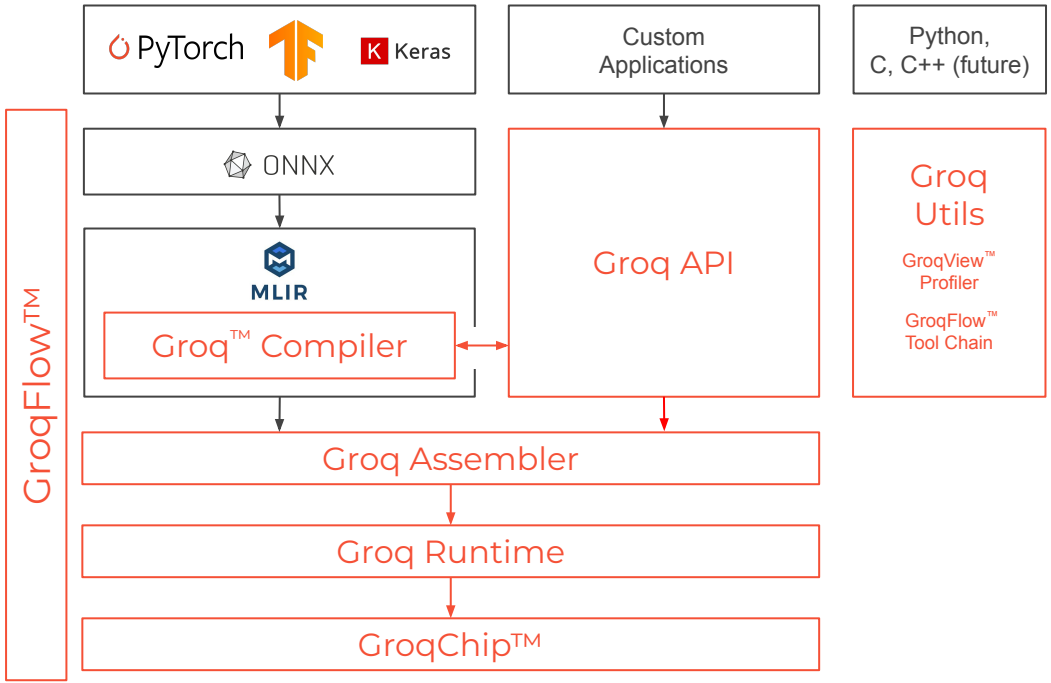
Traditional Non-deterministic Network



Software-scheduled Network

GroqWare™ Suite Intro

GroqWare™ Suite



DIVERSE SUITE OF DEVELOPMENT TOOLS

Out-of-Box

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

Fine Grained Control

Groq API provides finer grained control of GroqChip in order to support custom applications

+

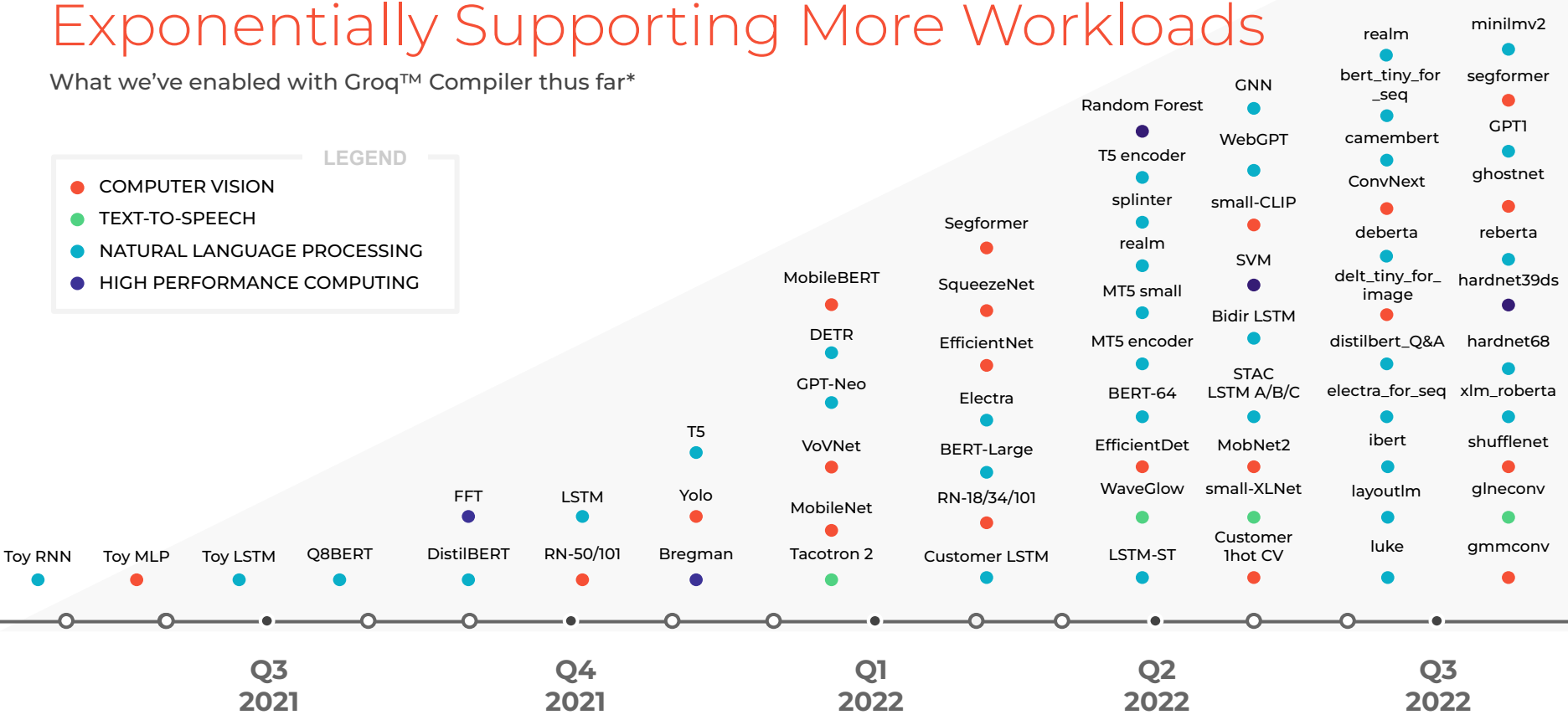
Productivity Tools

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

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

Hugging Face
Natural Language Processing

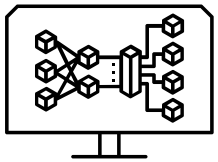
ONNX Model Zoo
Computer Vision

PyTorch Geometric
Graph Neural Networks

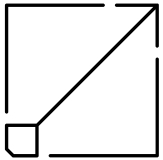
Hummingbird
Classical ML

WHERE IS GROQ GOING?

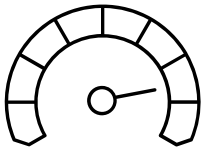
Our Vision



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

```python
import transformers
import torch
from groqflow import groqit

model = transformers.GPT2Model(transformers.GPT2Config())

inputs = {
    "input_ids": torch.ones(1, 1_024, dtype=torch.long),
    "attention_mask": torch.ones(1, 1_024, dtype=torch.float),
}

gmodel = groqit(model,inputs)

output = gmodel(**inputs)
```

*Gold standard of usability: off-the-shelf model from Huggingface.co

Introducing GroqFlow™

Step 2: Get some inputs

```python
import transformers
import torch
from groqflow import groqit

model = transformers.GPT2Model(transformers.GPT2Config())

inputs = {
    "input_ids": torch.ones(1, 1_024, dtype=torch.long),
    "attention_mask": torch.ones(1, 1_024, dtype=torch.float),
}

gmodel = groqit(model,inputs)

output = gmodel(**inputs)
```

*Gold standard of usability: off-the-shelf model from Huggingface.co

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)
12
13 output = gmodel(**inputs)
14
15
```

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

*Gold standard of usability: off-the-shelf model from Huggingface.co

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)
14
15
```

```
tensor([ 0.3628,  0.0489,  0.2952,  0.0022,
-0.0161, 0.3451, -0.3209, 0.0021, ...
```

*Gold standard of usability: off-the-shelf model from Huggingface.co

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™

GroqIt Key Functions

PYTORCH or ONNX

COMPILES models into Groq programs — gmodel = groqit(model,inputs)

EXECUTES programs on GroqChip™ — gmodel(**inputs)

BENCHMARKS programs on GroqChip (Coming soon) — latency = gmodel.benchmark()

GroqIt Key Functions

COMPILES models into Groq programs

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

EXECUTES programs on GroqChip™

```
gmodel(**inputs)
```

(No change!)

BENCHMARKS programs on GroqChip (Coming soon)

```
latency = gmodel.benchmark()
```

(No change!)

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

Notifications Fork 0 Star 1

Groq Inc. groq

4 repositories 1 member

ussions 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

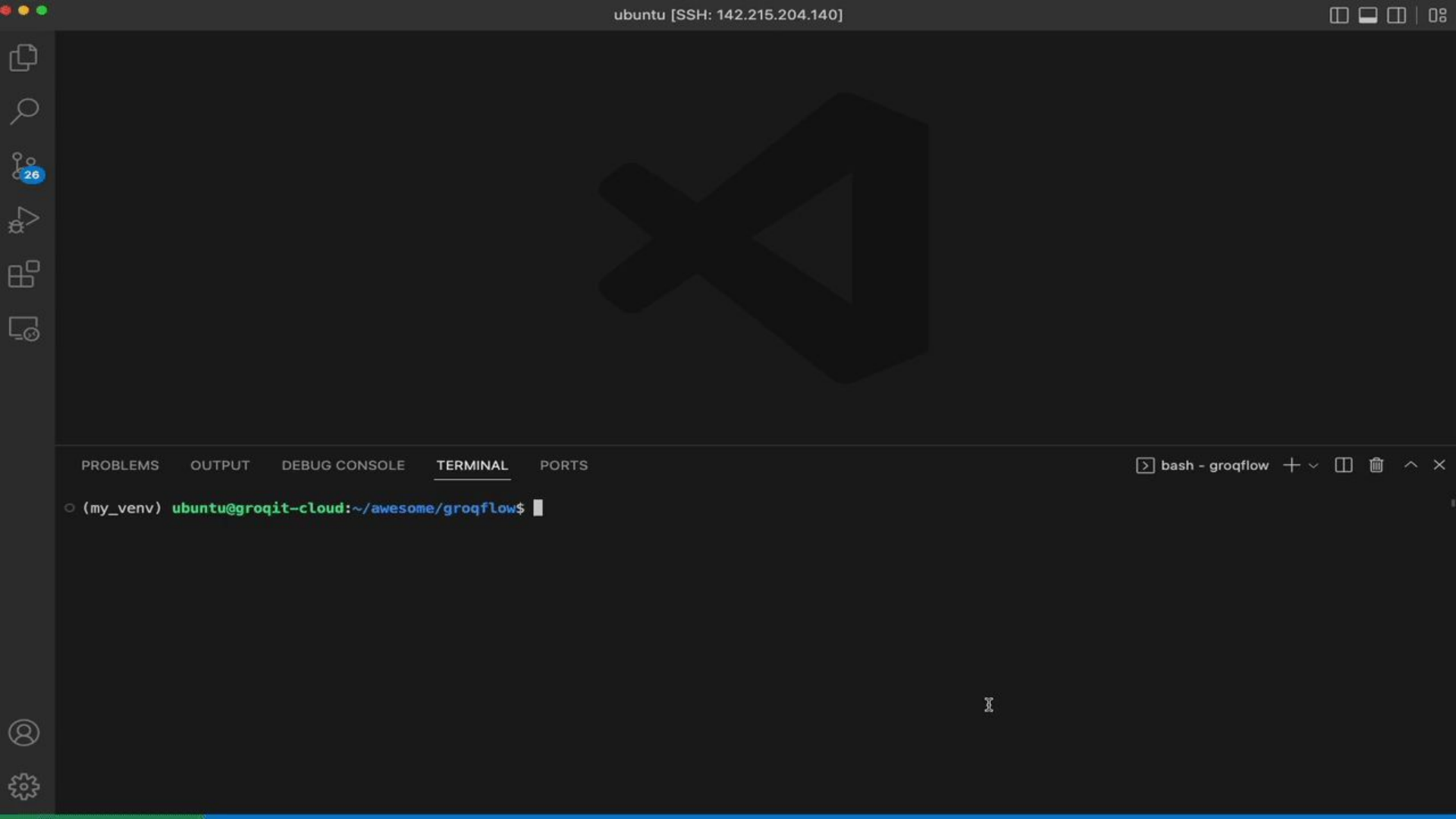
No releases published

Packages

No packages published

Languages

Python 100.0%


ubuntu [SSH: 142.215.204.140]
26
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
- PNAConv
- 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

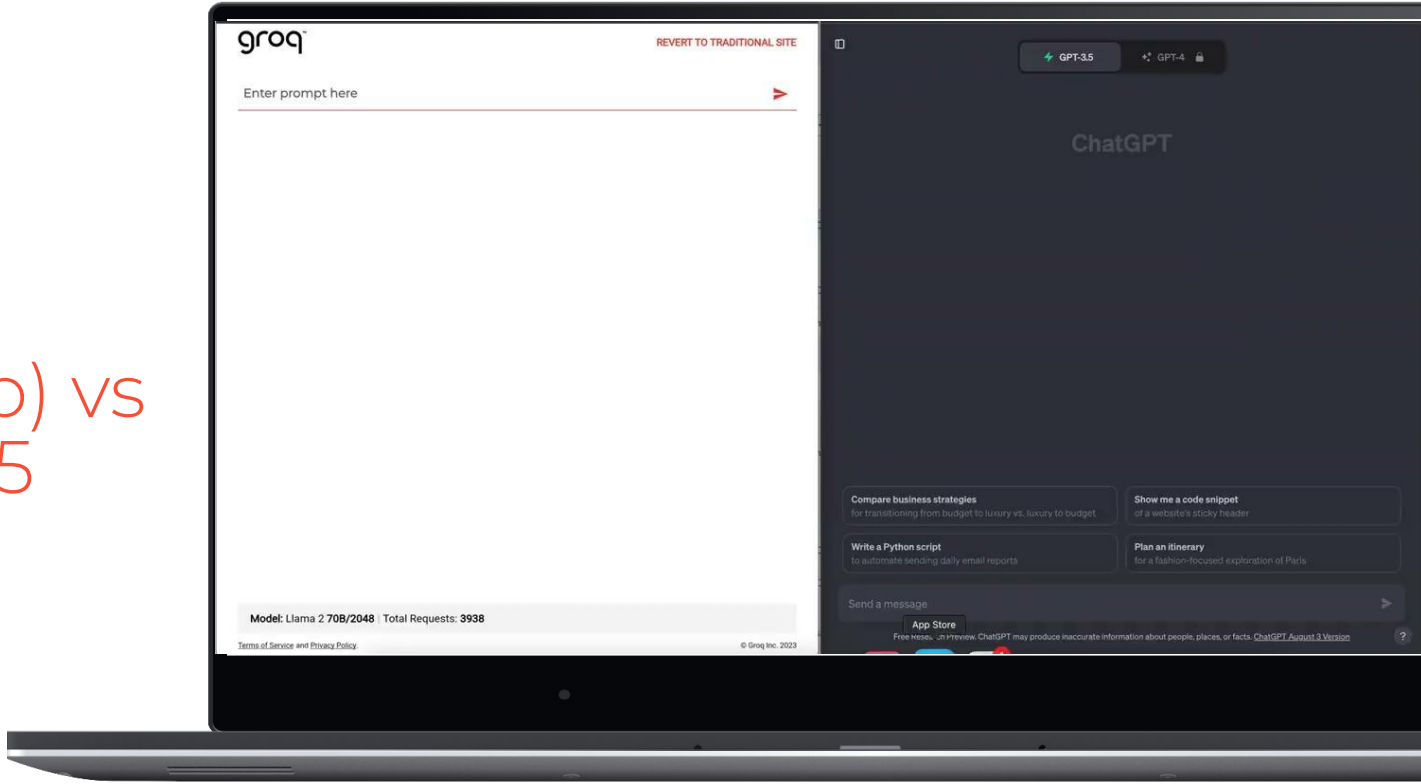
Others (Vision Models):

- 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

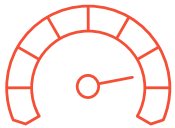
LLMs on Groq

Head-to-Head

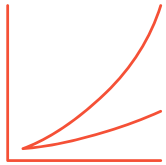
GroqChat (LLama 70b) vs ChatGPT 3.5 Turbo



Why We're Different

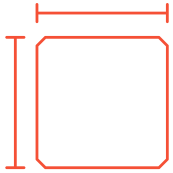


Performance



Pace

Predictability



Pinpoint

Come See Us
Booth 1681

SC23
Denver, CO | i am hpc.

+ groq™

Performance Disclaimers

[1] As measured at the host across 31 unique model inferences, at batch size 512, with 12,800 batches per run.

[2] As measured by comparing the max inference throughput that can be supported versus multiple on-chip co-resident models (GroqCard 1 vs Nvidia A100).

[3] Latest Gen HPC GPU (A100): 316 IPS @ 3ms, Batch 1, FP32; 46k IPS @5.5ms Batch 256, FP32
GroqCard™ 1 accelerator: 6.9k IPS @0.14ms, Batch 1, TruePoint™ 16; 193k IPS @0.6ms, Batch 128, TruePoint™ 16
As measured by comparing IPS at the fastest possible latency for both accelerators (GroqCard 1 vs A100)