ISC 2020
ML Hardware Workshop

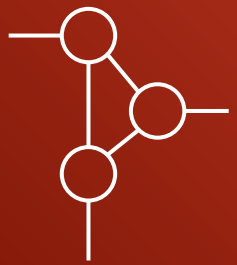
Groq
Dennis Abts - Chief Architect

Groq's Tensor Streaming Processor (TSP)

From Chips to Systems

groq

Motivation



Broadly speaking, **we are extracting as much computational density** --- measured in deep learning ops per floor tile in the data center.

This in turn drives the technology and implementation decisions to maximize the number of INT and FP arithmetic units (ALUs) and memory bandwidth necessary to feed them

Focus on batch-size 1 performance - drives technology and implementation decisions

Motivation

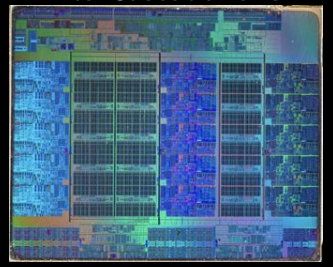
1. Uncompromised peak performance for batch-size 1 workloads.
2. Both integer AND floating-point numerics with a single-chip solution
3. Predictable performance through Determinism

Avoiding Complexity at the Chip Level

Conventional CPUs add features and add complexity:

- **Speculative execution** and out-of-order retirement
  - to improve instr level parallelism - **increases tail latency**
- **Implicit data flow** through cache memory hierarchies introduce **complexity and non-determinism**
  - **(e.g. DRAM → L3 → L2 → L1 → GPRs)** to hide DRAM access latency & pressure - **not energy or silicon efficient**

Intel Cascade Lake



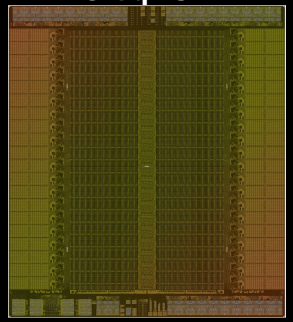
Requires dynamic profiling to understand execution time and throughput characteristics of deep learning models

The TSP simplifies data flow through Stream Programming:

- A large, single-level scratchpad SRAM - **fixed, deterministic latency**
- **Explicitly** allocate tensors in space and time unlocking massive memory concurrency, and compute flexibility along multiple dimensions:

  [device, hemisphere, memory slice, bank, address offset]

Groq TSP



Predictable Performance at Scale

groq

Avoiding Complexity at the System Level

Warehouse-scale Computers (WSCs) and Supercomputers

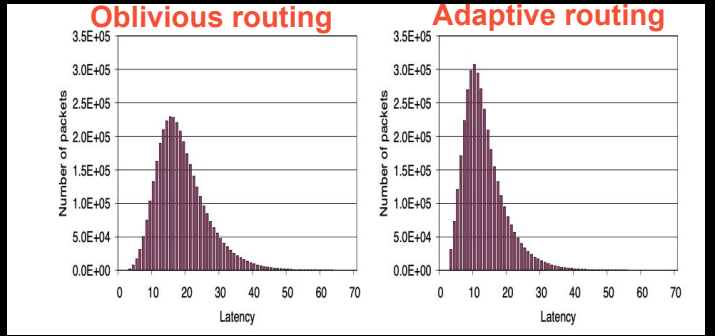
- Scaling to 20K+ nodes
- Increasingly heterogeneous (SmartNICs, CPU, GPU, FPGA)
- Latency variance limits application scale
- Related to diameter of the network
- Global adaptive routing is complex (out of order messages, faults, hotspots, route/load imbalance, congestion)

Software Defined Networking

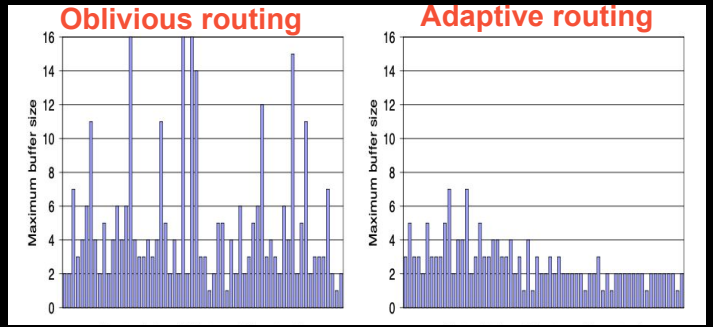
- Compiler controls traffic pattern in Groq C2C network
- Adaptive routing on chip to reduce variance on latency and reduce buffer occupancy

Extended with Existing Topologies Without Pitfalls

- High-radix switches to increase pin-bandwidth on each node/switch
- Low-diameter network topology (eg. Dragonfly, Flattened butterfly, HyperX)



Distribution of packet latency



Max buffer entries of middle stage switch in a 1K node network

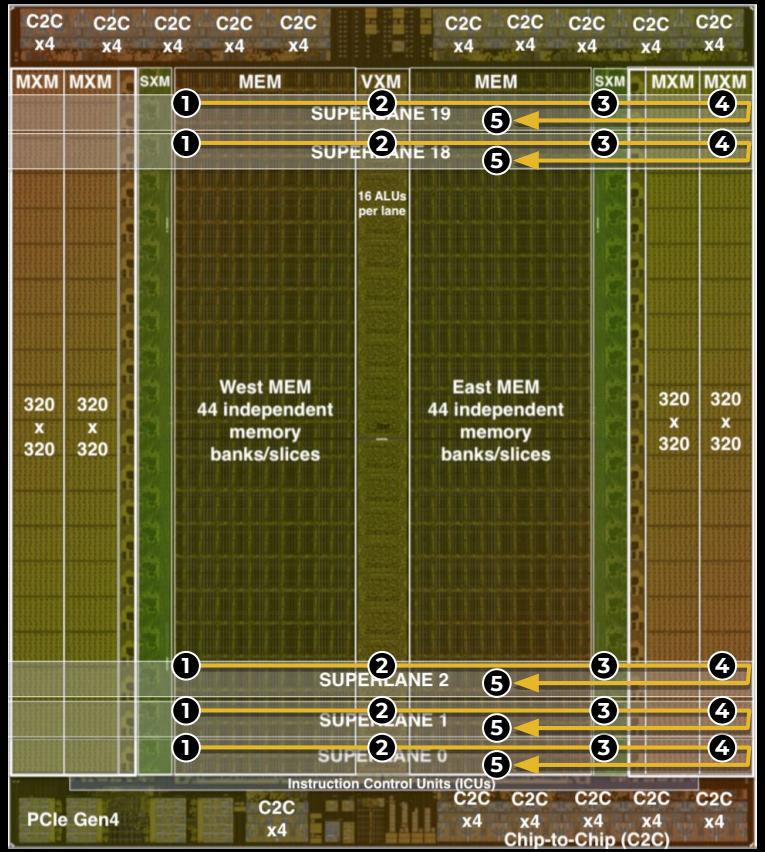
Tensor Streaming Processor Architecture

Chip to Chip Interconnect

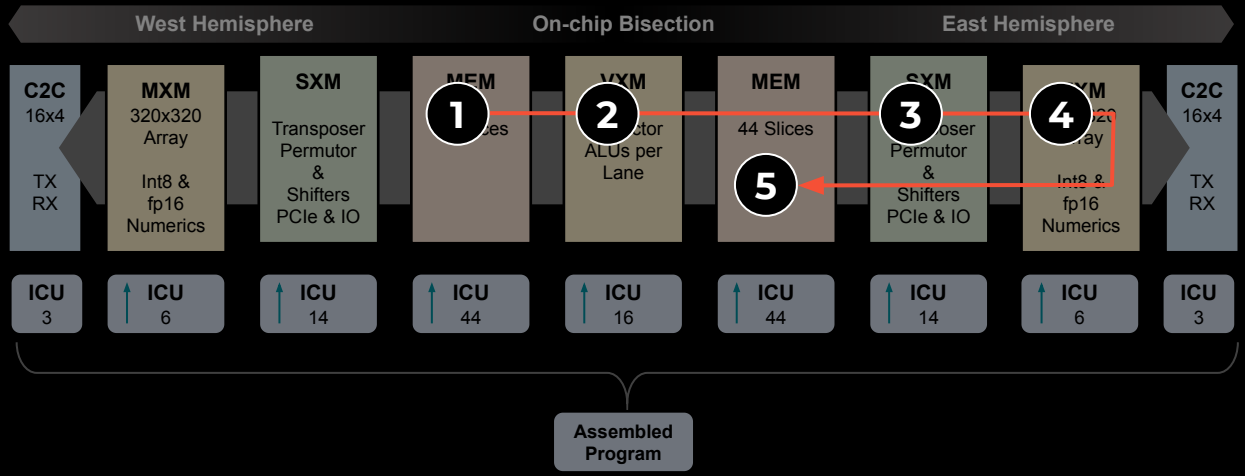
- Device ringed with **480 GB/s** off chip bandwidth

Simplified Compute Architecture

- Symmetric “hemispheres,” East and West with horizontal data flow along 20 superlanes
- The center VXM has 16 PEs per lane to support up to 5,120 total 32-bit computations per cycle (or 20k INT8 operations per cycle)
- The MXM has 320 lanes x 320 features: Stores 409,600 “weights” or parameters
  - **~750 Tops (Int8Acc32) peak**
  - **> 1 TeraOps/sec per mm²**



Dataflow Locality Within a "superlane"



One "row" of the on-chip 2D mesh is treated as a "superlane" and represents the minimum unit of computation.

Each superlane computes on a x16 SIMD unit. Taken together, the 20 Superlanes execute cycle by cycle to form a maxVL of 20x16=320 elements

Compiler has access to the following architecturally-visible state:

320-lane programming abstraction, each "superlane" implementing 16 lanes

144 independent instructions queues (ICUs) on-chip

64 logical streams - 32 Eastward, and 32 Westward

220 MiBytes of globally shared SRAM on-chip memory

SRAM read into a stream register sets it into motion, -- flowing toward the compute units like vector or matrix units, or data movement and reshapes in the SXM or a read/write to chip-chip network..

groq

Instruction Set

Low level, cycle accurate, 320 element vector ops, explicit resource selection

Function	Instruction	Description
ICU	`NOP N`	No-operation, can be repeated `N` times to delay by `N` cycles
	`Ifetch`	Fetch instructions from streams or local memory
	`Sync`	Parks at the head of the instruction dispatch queue to await barrier notification
	`Notify`	Releases the pending barrier operations causing instruction flow to resume
	`Config`	Configure low-power mode
	`Repeat n,d`	Repeat the previous instruction `n` times, with `d` cycles between iterations
MEM	`Read a,s`	Load vector at address `a` onto stream `s`
	`Write a,s`	Store stream `s` register contents into main memory address `a`
	`Gather s, map`	Indirectly read addresses pointed to by `map` putting onto stream `s`
	`Scatter s, map`	Indirectly store stream `s` into address in the `map` stream
	`Countdown d`	Set the delay `d` in cycles between loop iterations
	`Step a`	Set the stride `a` between subsequent generated memory addresses
	`Iterations n`	Set the loop bounds for address generation
VXM	unary operation	`z=op x` point-wise operation on 1 operand, `x`, producing 1 result, `z` (eg. `mask, negate`)
	binary operation	`z=x op y` point-wise operations with 2 operands *x* and *y* producing 1 result, (e.g. add, mul, sub)
	type conversions	Converting fixed point to floating point, and vice versa
	`ReLU`	Rectified linear unit activation function max(0,x)
	`TanH`	Hyperbolic tangent - activation function
	`Exp`	exponentiation e^n
	`RSqrt`	Reciprocal square root
MXM	`LW`	Load weights (LW) from streams to weight buffer
	`IW`	Install weights (IW) from streams or LW buffer into the 320 x 320 array
	`ABC`	Activation buffer control (ABC) to initiate and coordinate arriving activations
	`ACC`	Accumulate (ACC) either INT32 or FP32 result from MXM
SXM	`Shift` ***up/down N***	Lane-shift streams up/down by *N* lanes
	`Permute` ***map***	Bijective permute 320 inputs → `map` outputs
	`Distribute` ***map***	Rearrange or replicate data within a superlane (16 lanes)
	`Rotate` ***stream***	Rotate `n x n` input data to generate n^2 output streams with all possible rotations (`n`=3 or `n`=4)
	`Transpose sg16`	Transpose 16x16 elements producing 16 output streams with rows and columns interchanged
C2C	`Deskew`	Manage skew across plesiochronous links
	`Send`	Send a 320-byte vector
	`Receive`	Receive a 320-byte vector, emplacing it in main memory

groq

Data Type Support

Vector and Matrix compute units data types supported

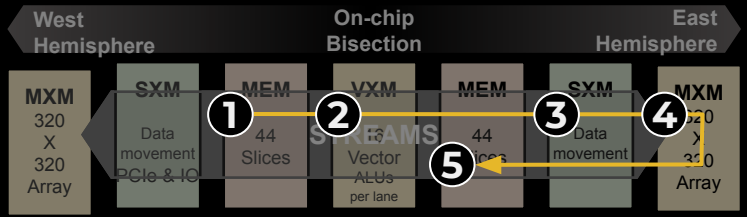
- Numerics for hardware-supported data types
- Matrix (MXM) unit supports INT8 and UINT8, and FLOAT16 (half-precision)
  - 320-element fused dot-product
  - INT32 or FP32 accumulation
- Vector (VXM) processor supports a superset of all data types

Data Type	Lowest Value	Highest Value
UINT8	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

Programming Models

- A **streaming programming model** provides a Producer-Consumer model where memory units are interleaved with compute units and ***streams*** provide the conduit among them
- Streams are constantly moving
- Stream architectures take advantage of dataflow inherent in the program

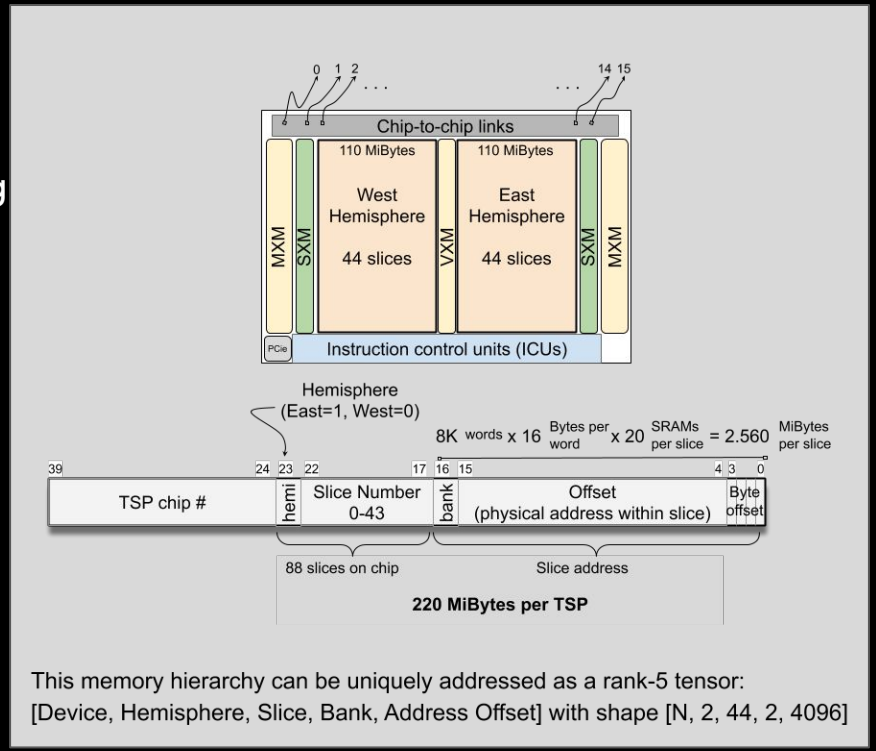
- Internet-scale applications are built using efficient programming models like Map-Reduce (Hadoop) on distributed parallel architectures



Partitioned Global Address Space

- Model parallelism across multiple TSPs in the node
- Memory allocation of tensors across multiple chips, with each memory allocation returning a 5-dim numpy array with fields corresponding to:
- [device, hem, memory slice, bank, address offset]

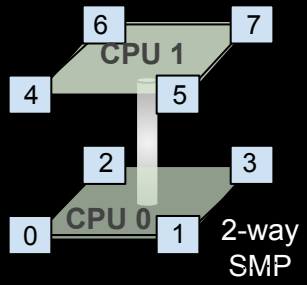
  Device within the system (0...65535)
  Hemisphere within the chip (East or West)
  Memory slice on within hemisphere (0..43)
  Bank within the slice (0 or 1) -- each slice has two internal banks for concurrency
  Address offset (0..8191) -- each SRAM stores 8K 16-byte words



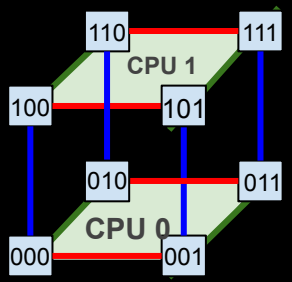
This memory hierarchy can be uniquely addressed as a rank-5 tensor: [Device, Hemisphere, Slice, Bank, Address Offset] with shape [N, 2, 44, 2, 4096]

Flexible Node Organization

Flexible software-defined networking allows deployment choices



Eight Single TSP or
Two 4-way TSP groups



One 8-way TSP group

Node	Scale	Description
8x1	320x1	Eight single-chip TSPs
2x4	80x4	Two 4-way TSP groups
1x8	40x8	One 8-way TSP group

Strong-scaling across TSPs within node

Model Parallelism and Data Parallelism

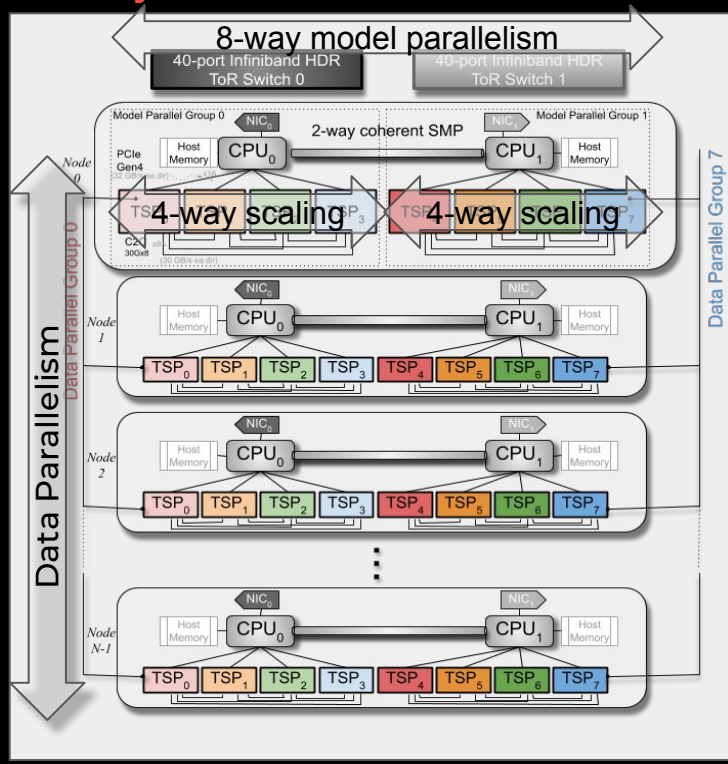
Exploiting both model parallelism and data parallelism across the system

Model Parallel

- Model layers are distributed across 4-8 TSPs in a node
- 4-TSPs are fully connected with Groq chip-to-chip (C2C) links for fine-grain synchronous messaging
- 8-TSPs share a coherent memory domain in 2-way SMP host memory
- Allows strong-scaling across 4 or 8 TSPs within a shared memory node

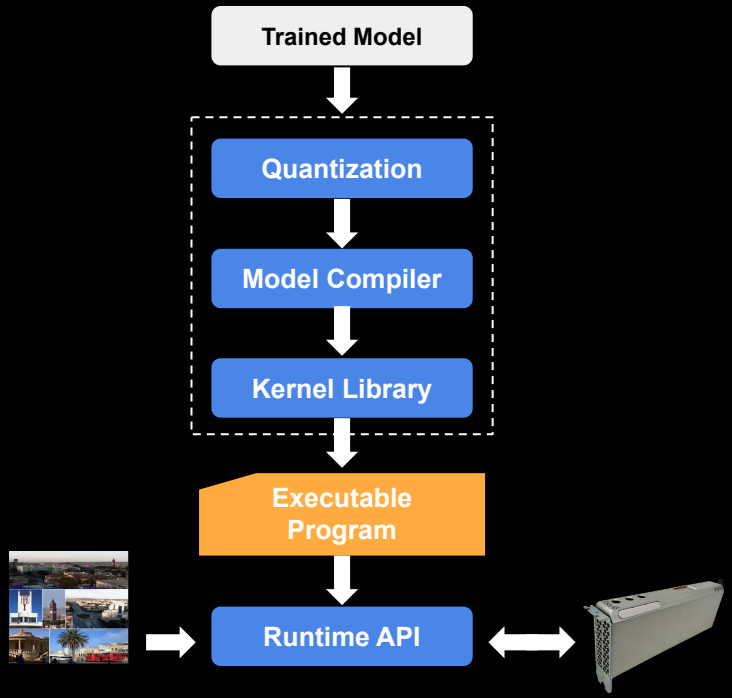
Data Parallel

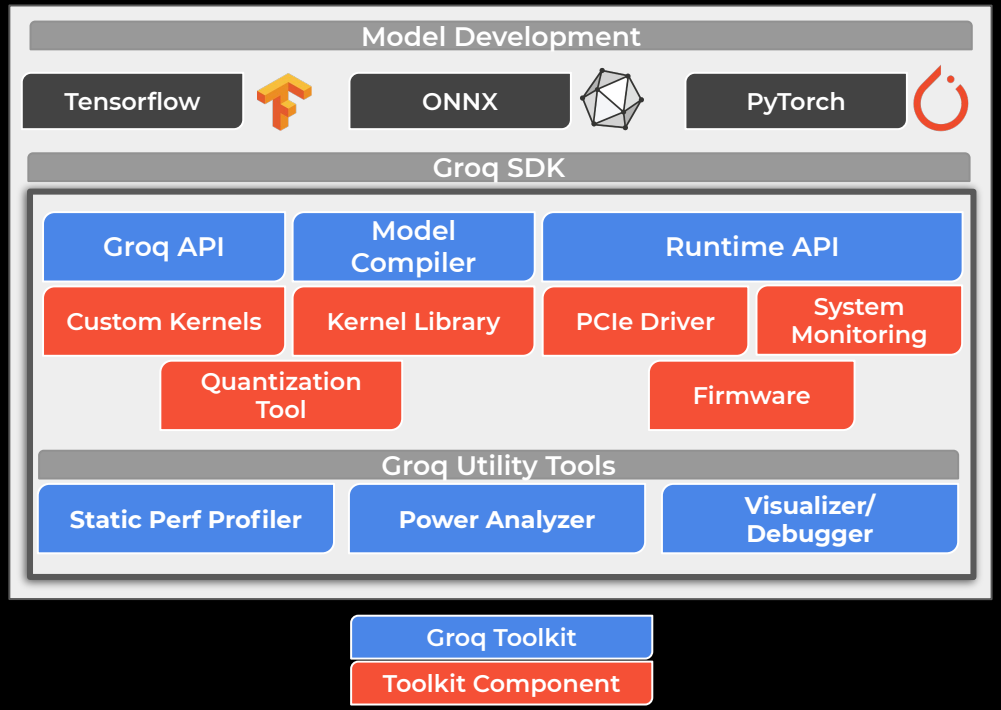
- Data parallelism across nodes
- TSPs of the same color share the same set of weights
- **MPI or MapReduce** used to update/distribute weights of TSPs within the same Data Parallel Group (same color)



Groq Developer Software Stack

A suite of development, validation, and inference execution tools





Groq API - Direct Stream Programming

Direct-to-architecture API for controlling Groq hardware

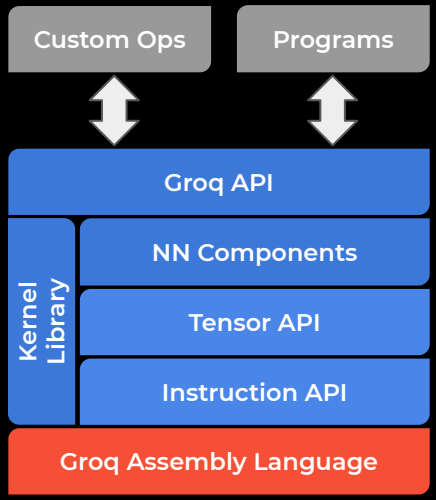
- Direct access to hardware resources (functional units, streams, memory)
- Full control over program execution and timing
- Built on top of the Groq assembly language for easier programmability

Tensor-level abstractions provide building blocks for reusable components and kernels

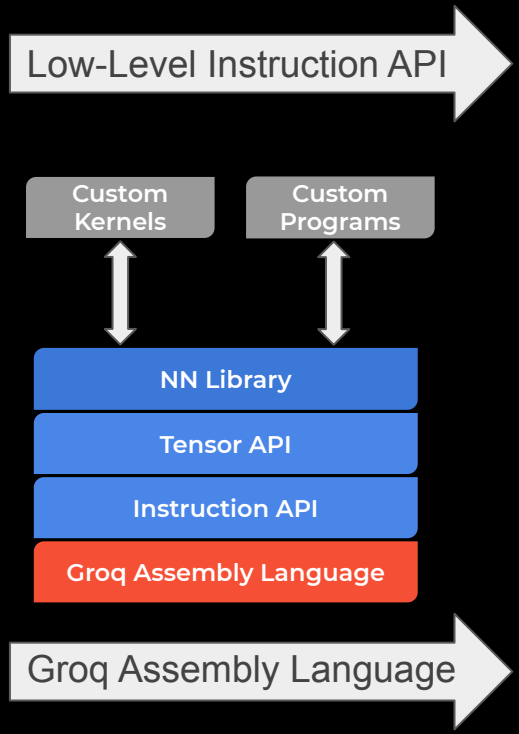
- Operations over multi-rank tensors
- Examples: Pointwise operations, accumulation, dataflow

Library of reusable kernels and neural network components

- Extensible using the Groq API
- Building blocks for constructing kernels and programs



Groq API



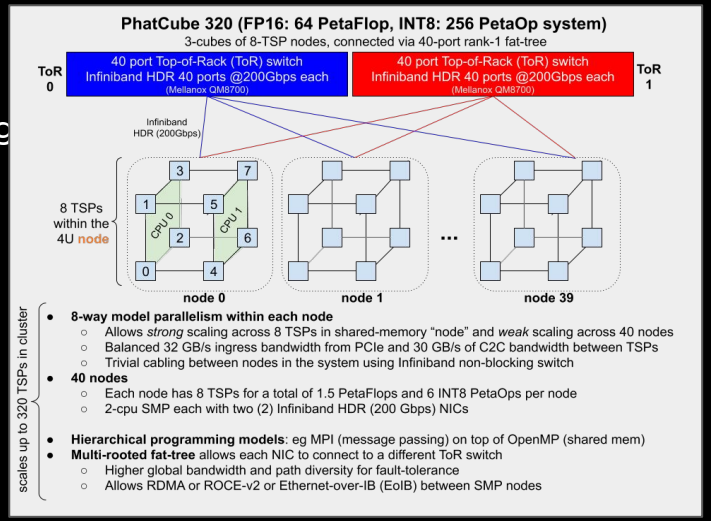

```python
import groq.instruction as inst

c1 = inst.constant("W1-0", [1, 2, 3, 4], comment="input x")
c2 = inst.constant("W2-0", [4, 3, 3, 4], comment="input y")

x = inst.read(c1, "S_0_e")
y = inst.read(c2, "S_4_e")

sg_z = inst.vxm_binary(0, inst.VXMOp.ADD_SAT, inst.VXMFormat.INT8,
                       x, y, "SG4_0_w")
out_addrs = [inst.parse_address('W3-0')]

w = inst.write(out_addrs[0], sg_z[0], time=12)
inst.output_section(inst.IOFormat.INT8, 4, out_addrs)

inst.print_groq_assembly()
```

```
.unit Mem West 1
// Mem West 1
.read 0
0:     read 0, S_0_eastward
.unit Mem West 2
// Mem West 2
.read 0
0:     read 0, S_4_eastward
.unit Mem West 3
// Mem West 3
.write 0
12:    write 0, S_0_westward
.unit VXM 0
// VXM 0
6:     addsat int8, SG4_0_eastward, SG4_1_eastward, SG4_0_westward
```

Scaling from Chips to Systems

Multiprocessor TSPs using direct networks

- Provide a globally synchronous software abstraction of the multi-TSP system
- Software-defined networking allows us to load balance links in the system
- Example systems scaling from 1 TSP to 320 TSPs with strong scaling across the 8 TSPs within each node, and weak scaling (eg. mini-batch) across 40 nodes today
- **Direct networks** within the SMP node
  - 3D torus
  - N-dim binary hypercube
  - Generalized hypercube (aka Flattened Butterfly)
- Hierarchical programming models
  - OpenMP (threads) within the shared memory node with up to 8 TSPs
  - Distributed memory message passing (MPI or MapReduce/Hadoop) between nodes for weak scaling



The Groq Box

Dense Compute Platform

- 6 PetaOP/s, 240GBps Interconnect
- Contains 8 Groq TSP 100 Cards in a 4U chassis
- Configured as 8 cards, 2x4 cards or 8 interconnected cards
- DP AMD ‘Rome’ server with 160 PCIe lanes
- 3.3kWs of Power

Multi-Box Scalability

- Supports in box model parallelism
- Data parallelism across multiple boxes
- Scales up to arrays of up to 320 TSP100 cards
- 200G Ethernet or Infiniband HBR connections
- User-level runtime API for low-latency to host , Groq API support

Scale Configurations

Solutions for low-latency, compute-intensive applications

Node Scale - 2x 4-way or 1x 8-way model parallelism within each node

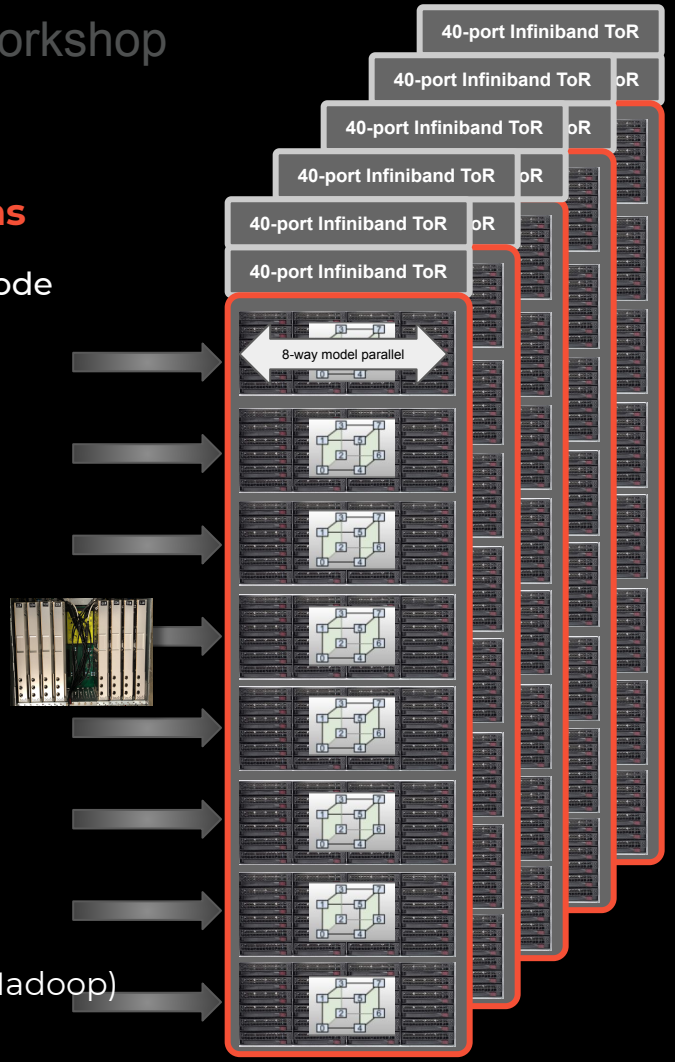
- **6 PetaOps (INT8); 1.5 PetaOp (FP16)**
- 240 GB/s bisection bandwidth (strong scaling local to node)

Rack Scale - Up to 64 TSPs in a rack

- **48 PetaOps (INT8); 12 PetaFlops (FP16)**
- Two 200 Gbps Infiniband HDR links to ToR 40-port switches
  - 50 GB/s bisection weak scaling bandwidth between nodes(minibatch parallel)

System Scale - Scales to 100s of Nodes in a system

- **E.g. 240 PetaOps INT8; 60 PetaFlops FP16 (320 TSP system)**
  Worst-case **system latency of 2.5 µsec**
  - Directly impacts **AllReduce** latency and strong scaling
- Hierarchical programming model:
  - Threads within SMP node (eg pthreads, OpenMP)
  - Message-passing between nodes (eg MPI, MapReduce/Hadoop)



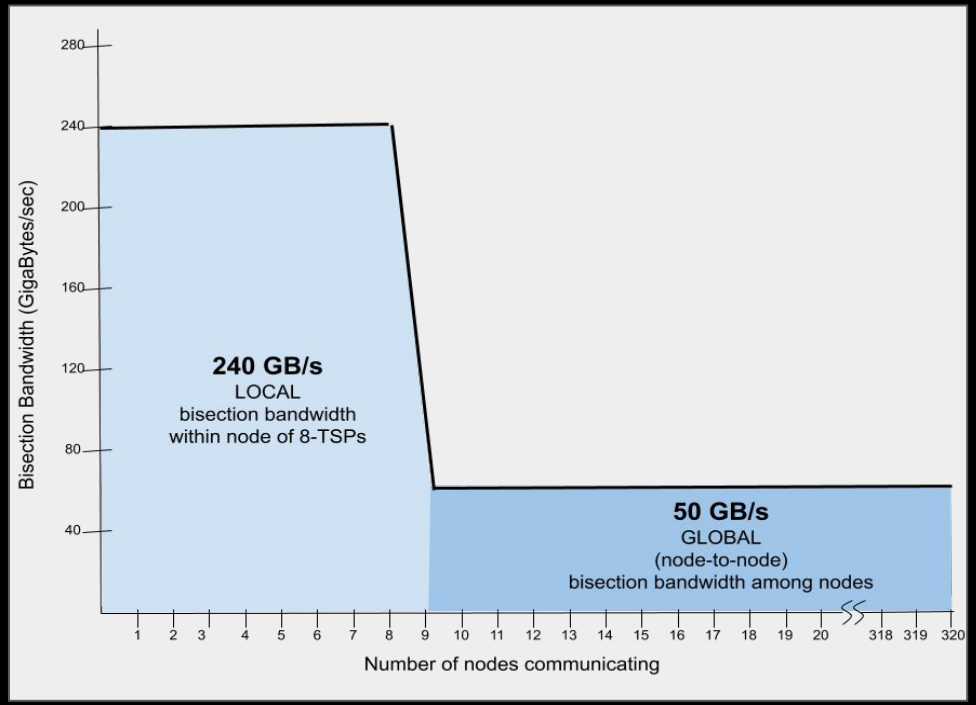
groq

Bandwidth profile

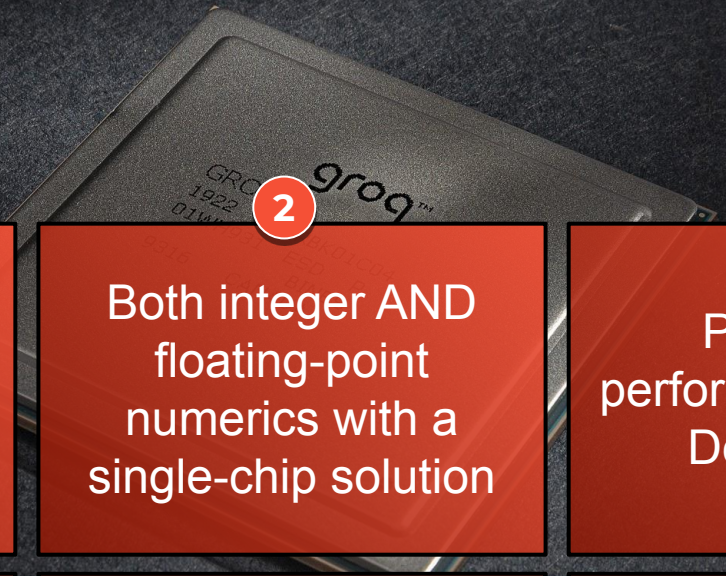
Exploiting locality at different levels of the packaging hierarchy

- Dense system packaging allows for 8 TSPs in a 4U chassis with 2-way SMP CPU
- **LOCAL BANDWIDTH**: Strong scaling across the TSPs within each node with 240 GB/s bisection bandwidth communicating between TSPs in node
- **GLOBAL BANDWIDTH**: Weak scaling across the 40 nodes in system using distributed memory message passing. 50 GB/s of global bisection bandwidth

Tremendous compute density allows Groq systems to do more computation using a smaller network diameter yielding low system latency.



Putting it all together



1	2	3
Uncompromised peak performance for batch-size 1 workloads.	Both integer AND floating-point numerics with a single-chip solution	Predictable performance through Determinism
Outperform Existing Solutions	*Application Versatility*	*Performance Scalability*

Simplify Compute, Everywhere.