Software-defined Hardware with Groq's Tensor Streaming Processor

A hardware-software approach

HELLO ATPESC

Andrew Ling, Ph.D.

Sr. Director Software

MY GOAL TODAY

Introduce you to Groq's approach to SW defined hardware in the era beyond CPUs

Details...

Appears in the Proceedings of the International Symposium on Computer Architecture (ISCA-47), June 1, 2020.

Think Fast: A Tensor Streaming Processor (TSP) for Accelerating Deep Learning Workloads

Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, Max Baker, Tom Hawkins, Andrew Bell, John Thompson, Temesghen Kahsai, Garrin Kimmell, Jennifer Hwang, Rebekah Leslie-Hurd, Michael Bye, E.R. Creswick, Matthew Boyd, Mahitha Vinigalla, Evan Laforge, Jon Purdy, Purushotham Kamath, Dinesh Maheshwari, Michael Beidler, Geert Rosseel, Omar Ahmad, Gleb Gagarin, Richard Czekalski, Ashay Rane, Sahil Parmar, Jeff Werner, Jim Sproch, Adrian Macias, Brian Kurtz

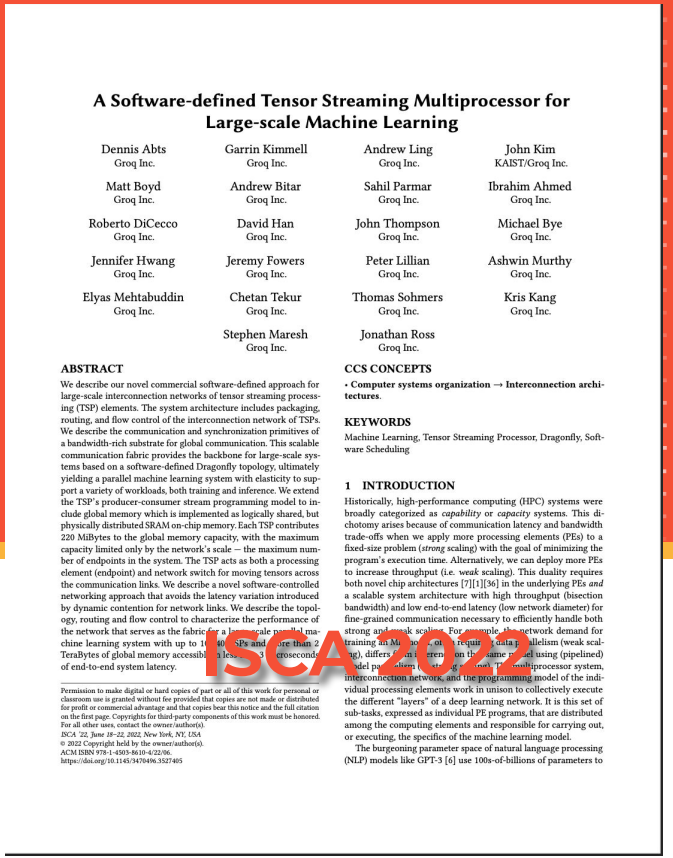
Groq, Inc. · Mountain View, California

Abstract—In this paper, we introduce the Tensor Streaming Processor (TSP) architecture, a functionally-sliced microarchitecture with memory units interleaved with vector and matrix deep learning functional units in order to take advantage of dataflow locality of deep learning operations. The TSP is built based on two key observations: (1) machine learning workloads exhibit abundant data parallelism, which can be readily mapped to tensors in hardware, and (2) a simple and deterministic processor with producer-consumer stream programming model enables precise reasoning and control of hardware components, achieving good performance and power efficiency. The TSP is designed to exploit parallelism inherent in machine-learning workloads, including instruction-level parallelism, memory concurrency, data and model parallelism, while guaranteeing determinism by eliminating all reactive elements in the hardware (e.g. arbiters, and caches). Early ResNet50 image classification results demonstrate 20.4K processed images per second (IPS) with a batch-size of one—a 4× improvement compared to other modern GPUs and accelerators [44]. Our first ASIC implementation of the TSP architecture yields a computational density of more than 1 TeraOp/s per square mm of silicon for its 25×29 mm 14nm chip operating at a nominal clock frequency of 900 MHz. The TSP demonstrates a novel hardware-software approach to achieve fast, yet predictable, performance on machine-learning workloads within a desired power envelope.

I. Introduction

The world is increasingly turning to computationally-intensive deep learning algorithms to solve important problems in science, transportation, security, and beyond. These workloads continue to grow both in size and complexity, presenting serious scalability, performance, and usability challenges for traditional CPU and GPU architectures. Knowing that ALUs and utilize them ... many microarchitectures ... program execution. Unfo... ments such as caches, branch predictors, and prefetchers help runtime stalls. Furthermore, ... worst-case performance. Over the last decade, data center operators have installed many-core systems as a stalwart fixture in warehouse-scale computers (WSC) [7], [11], [13]. These increasingly heterogeneous systems have dozens of processing cores that vary widely in form and function, including GPUs, TPUs, FPGAs, and smart IO controllers for efficient remote direct memory access. Such efforts have largely focused on accelerating deep neural network *training* and *inference* performance for a wide range of workloads: convolutional neural networks (CNNs) for recommendation algorithms, image classification, ... natural language processing, computer vision (CV) and ... networks (RNNs) for ... [25], and now attention and ... models has been the catalyst for ... architectural innovation [21]. The demand for domain-specific architectures, as well as their widespread adoption into cloud computing ecosystems [34], [48], provides a unique opportunity for novel microarchitectures targeting deep learning applications. A range of innovative solutions are emerging—from Cerebras' wafer-scale integration [15] to more traditional chip multiprocessors like GraphCore's IPU [25]. Broadly speaking, all of these efforts are focused on delivering more compute capability per floor tile in the data center. In other words, the focus is on

Fig. 1. Conventional 2D mesh of cores (a) reorganized into a functionally sliced arrangement of tiles (b).



A Software-defined Tensor Streaming Multiprocessor for Large-scale Machine Learning

Dennis Abts, Groq Inc.
Garrin Kimmell, Groq Inc.
Andrew Ling, Groq Inc.
John Kim, KAIST/Groq Inc.
Matt Boyd, Groq Inc.
Andrew Bitar, Groq Inc.
Sahil Parmar, Groq Inc.
Ibrahim Ahmed, Groq Inc.
Roberto DiCecco, Groq Inc.
David Han, Groq Inc.
John Thompson, Groq Inc.
Michael Bye, Groq Inc.
Jennifer Hwang, Groq Inc.
Jeremy Fowers, Groq Inc.
Peter Lillian, Groq Inc.
Ashwin Murthy, Groq Inc.
Elyas Mehtabuddin, Groq Inc.
Chetan Tekur, Groq Inc.
Thomas Sohmers, Groq Inc.
Kris Kang, Groq Inc.
Stephen Maresh, Groq Inc.
Jonathan Ross, Groq Inc.

ABSTRACT

We describe our novel commercial software-defined approach for large-scale interconnection networks of tensor streaming processing (TSP) elements. The system architecture includes packaging, routing, and flow control of the interconnection network of TSPs. We describe the communication and synchronization primitives of a bandwidth-rich substrate for global communication. This scalable communication fabric provides the backbone for large-scale systems based on a software-defined Dragonfly topology, ultimately yielding a parallel machine learning system with elasticity to support a variety of workloads, both training and inference. We extend the TSP's producer-consumer stream programming model to include global memory which is implemented as logically shared, but physically distributed SRAM on-chip memory. Each TSP contributes 220 MiBytes to the global memory capacity, with the maximum capacity limited only by the network's scale — the maximum number of endpoints in the system. The TSP acts as both a processing element (endpoint) and network switch for moving tensors across the communication links. We describe a novel software-controlled networking approach that avoids the latency variation introduced by dynamic contention for network links. We describe the topology, routing and flow control to characterize the performance of the network that serves as the fabric for a large-scale parallel machine learning system with up to 10[illegible] SPs and [illegible] are than 2 TeraBytes of global memory accessible [illegible] 3 [illegible] crosecond of end-to-end system latency.

ISCA '22, June 18–22, 2022, New York, NY, USA

ACM ISBN 978-1-4503-8610-4/22/06.
https://doi.org/10.1145/3470496.3527405

Groq in a Nutshell

Founded	2016
Flagship Product	Groq Tensor Streaming Processor Chip (TSP) and Software for use in AI, Machine and Deep Learning applications
Target Market(s)	AI and Machine Learning Hyperscalers, Government, HPC clusters, Autonomous vehicles, high performance edge appliances
Employees	~250
Founder/CEO	Jonathan Ross

ML Compilers and Tools

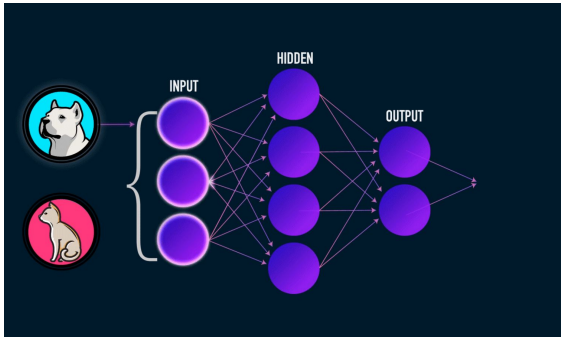
DNN Library Developers

Host-IO Developers

ASIC/HW Designers

Supporting Workloads of ***Today***

- What would you build given the data flow nature of Machine Learning Workloads and HPC?
  - "Nodes" in the computational graph represent **operators** and "edges" are the **operands** and **results**.



- Groq re-examined the hardware-software contract
- Created hardware that is much more predictable and streaming based
- ***Gave more control to software!***

Have we been here before? RISC ➔ VLIW?

With RISC, CPU architecture hides a lot of complex control logic from software

Caching, prefetching, out-of-order execution, branch prediction

- Area cost of this control hardware reduces available area for raw compute

Very Long Instruction Word (VLIW) to the rescue?

The next natural step in the trend of shifting work and control from hardware to software

- Execution ordering, data prefetching, branch resolution
- Compiler responsible for exploiting instruction-level parallelism (ILP) available in a program

Itanium…

Compiler struggled to achieve good ILP due to limited view of dynamic hardware behaviour

- Cache misses, dynamic mem dependencies, dynamic branches

A New Golden Age in Computing

The end of CPU hegemony

Growing demand in dataflow dominated compute

Slowing of Moore's Law and Dennard Scaling

CPU "abstraction" no longer the only foundation for developing software

Dawn of a new golden age in computing

Hennessy & Patterson: "A New Golden Age for Computer Architecture"

Lattner: "A New Golden Age of Compilers"

Karpathy: "Software 2.0" → A New Golden Age for Algorithms

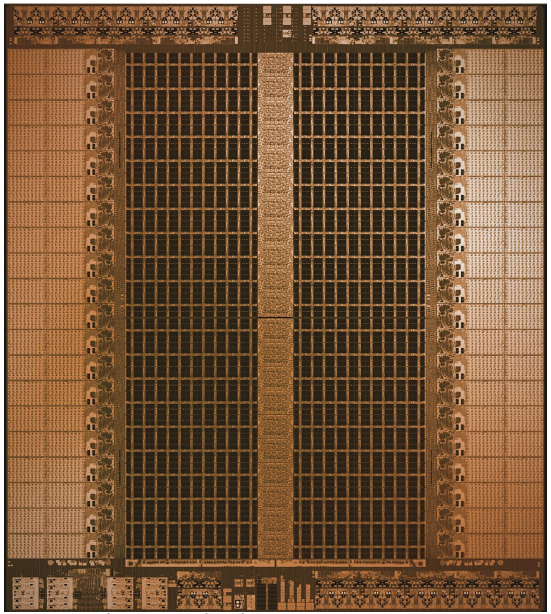
In the past, hardware (CPUs) has defined our software

The hardware-software abstraction contract has been reopened!

Opportunity now to achieve "software-defined hardware"

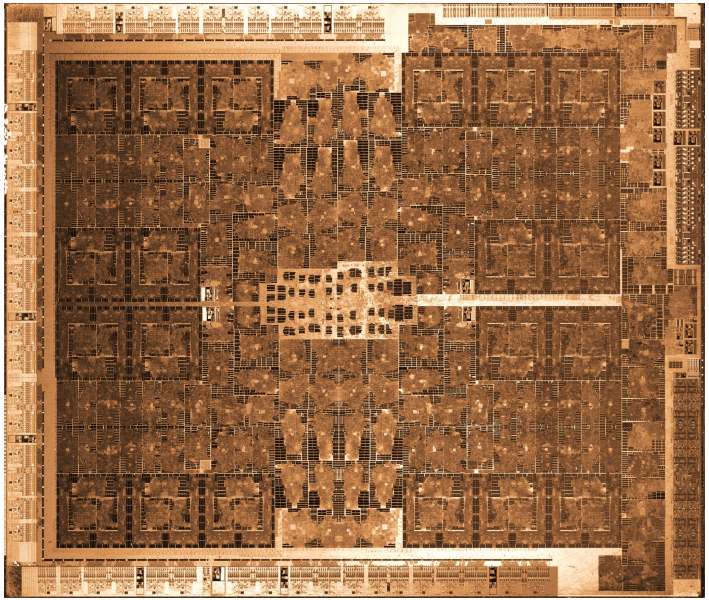
Simplifying Compute

GroqChip™



Simplicity enables
Compute Performance

Nvidia GTX 1070



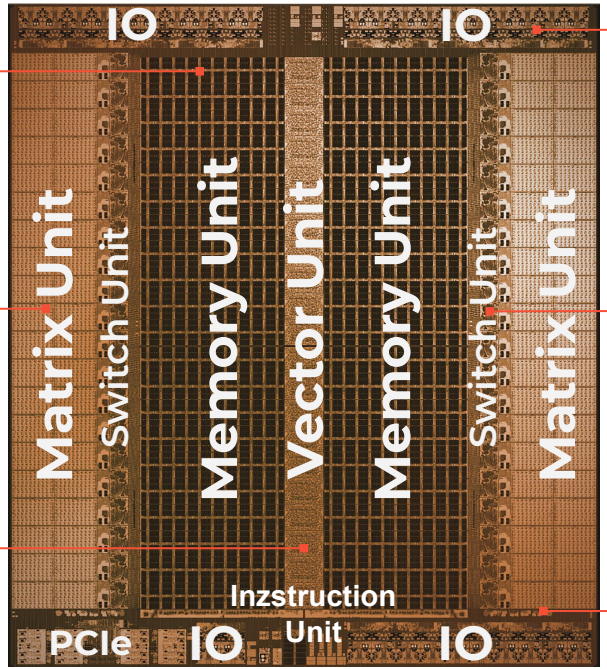
Complexity leads to more
Compute Costs

GroqChip Scalable Architecture

220MB SRAM
80 TB/s of Stream Bandwidth on-chip
Massive concurrency

Dense MatMul
720 TeraOP/s (1 TeraOP / s / mm^2)

Vector Units



480GBps Chip-2-Chip Links

Dataflow

Instruction Control

Designing for determinism…

- Design choices along the way need to accommodate the “design for determinism” design philosophy

- Hardware must enable the compiler and runtime interfaces to reason about program execution
  - Memory consistency model must be well understood - disallowing memory references from being reordered
  - No “reactive components” like arbiters, crossbars, replay mechanisms, caches, etc
  - Software must have access to the **architectural-visible machine state** in order to intercept the data (operands) with the instruction that will execute on them.

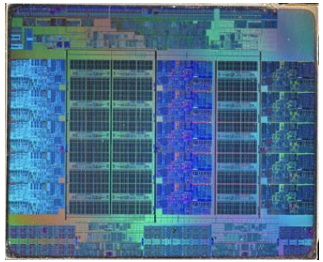
- Compiler “knows” the exact location of every tensor on-chip

- In this way, the compiler is *orchestrating* the arrival of operands and the instructions which use them. The **producer-consumer stream programming model** allows a set of “streaming register files” to track the state of each tensor flowing through the chip.

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**

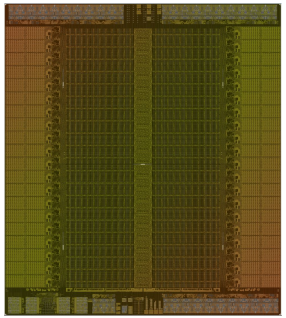


Requires dynamic profiling to understand execution time and throughput characteristics of deep learning models

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:

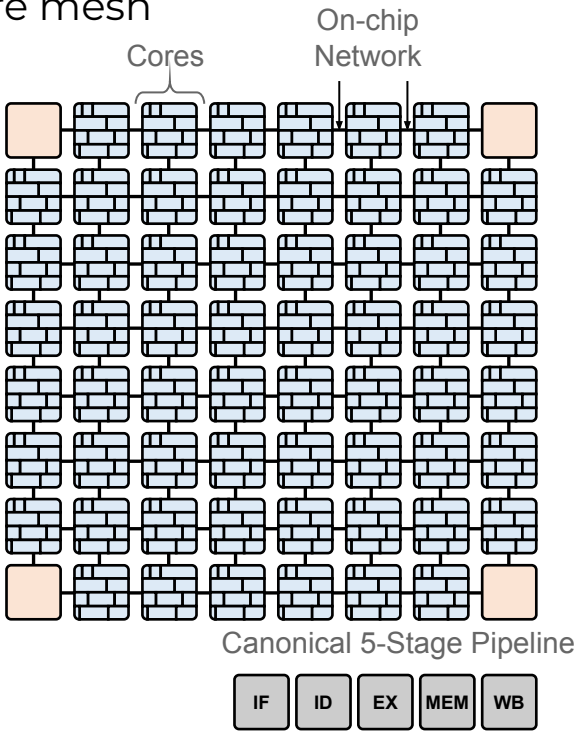
  [device, hemisphere, memory slice, bank, address offset]



Predictable Performance at Scale

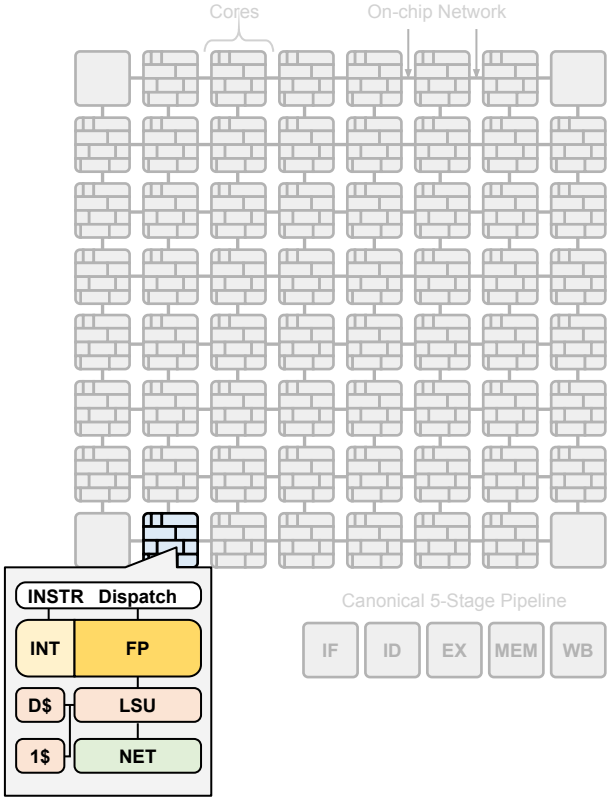
Functionally Sliced Microarchitecture

Reorganizing the multicore mesh



Functionally Sliced Microarchitecture

Reorganizing the multicore mesh



Functionally Sliced Microarchitecture

Reorganizing the multicore mesh

Reorganize a conventional manycore 2D mesh

MEM: on-chip SRAM

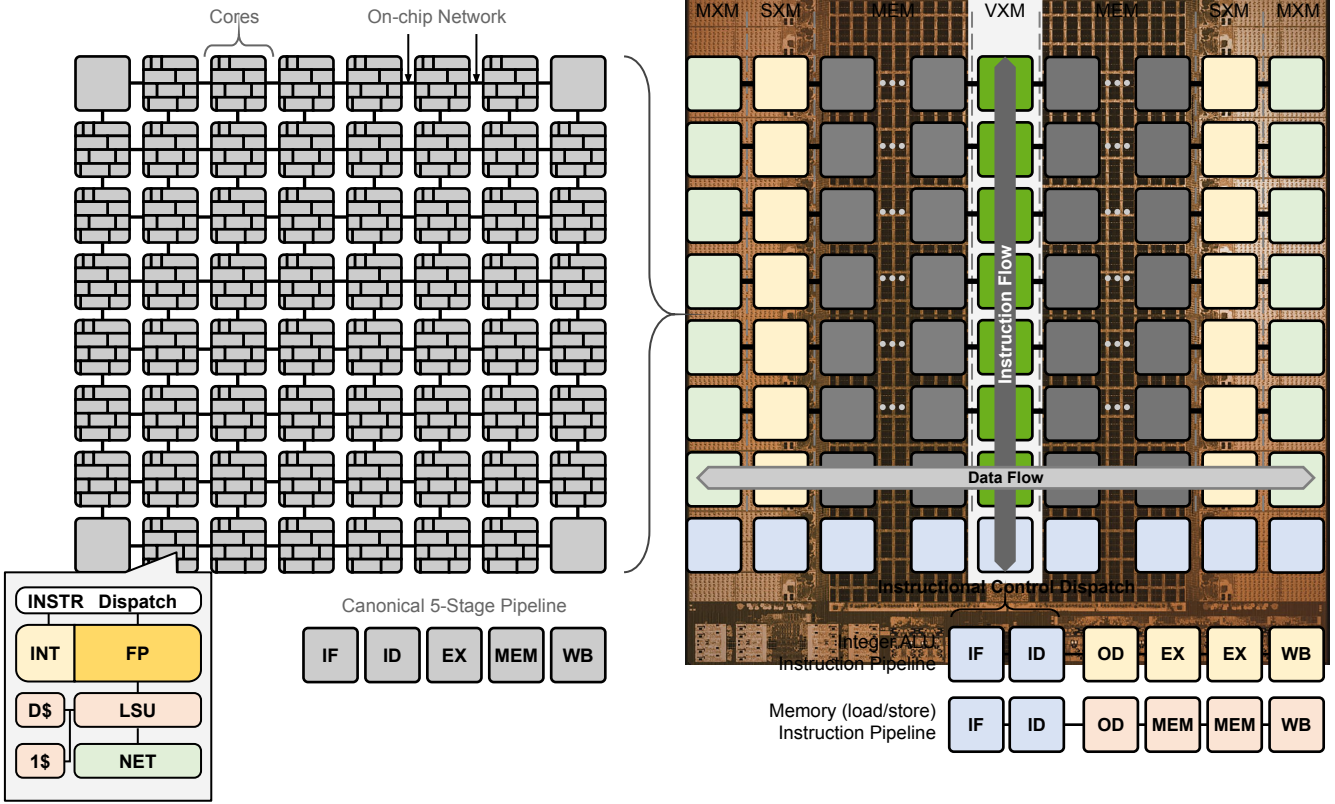
VXM: vector unit

MXM: matrix unit

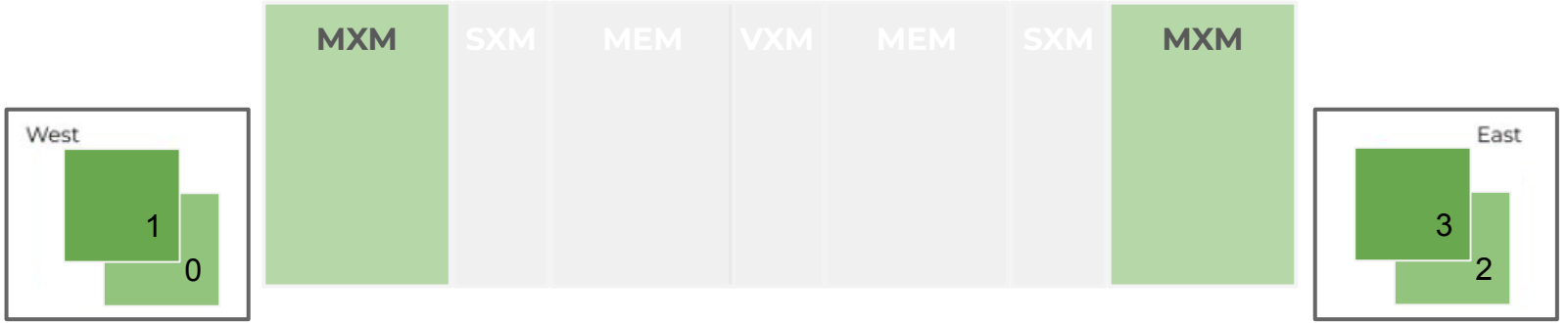
SXM: data reshapes

Tensor operands and results flow on **"streams"** horizontally

Instructions flow vertically executed in a SIMD manner



MXM - Matrix Multiply Engines



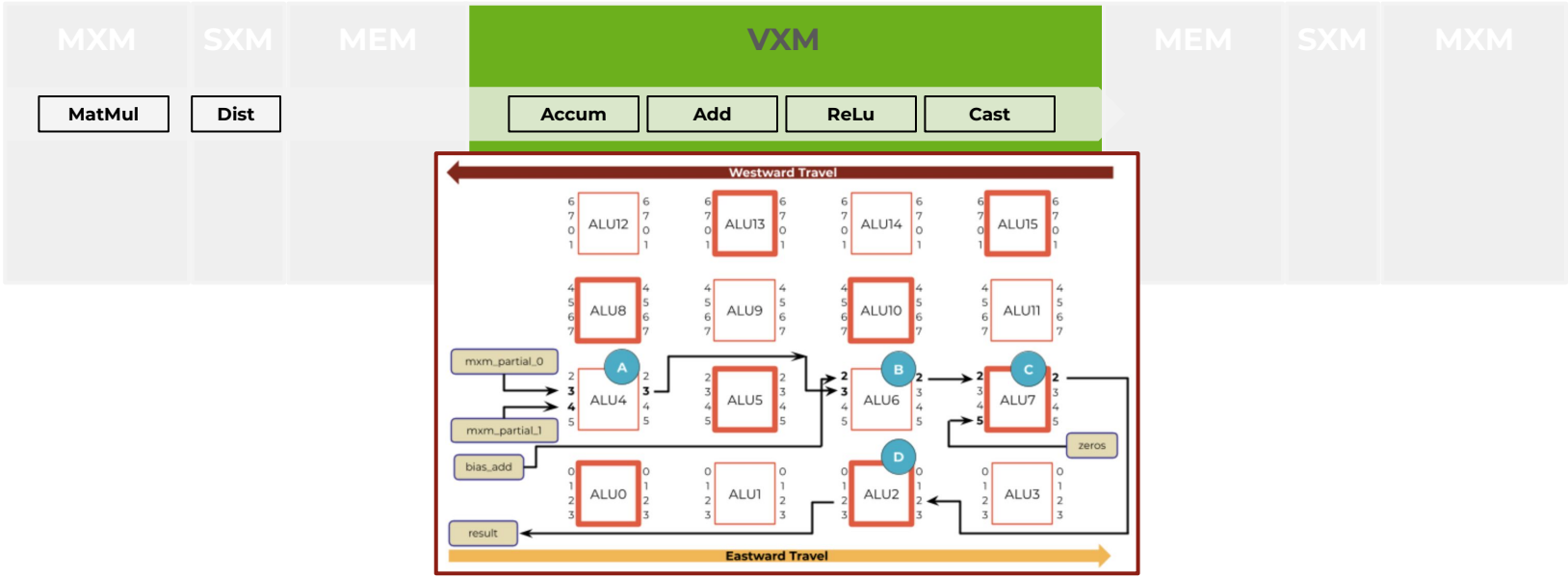
Numeric Mode	Max Size	Supported Density	Result Tensor
int8	[1, 320] x [320, 320]	Two per MXM	int32
float16	[1, 320] x [160, 320]	One per MXM	float32

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 and Complex, Customized Functions



Dataflow begins with memory Read onto Stream Tensor

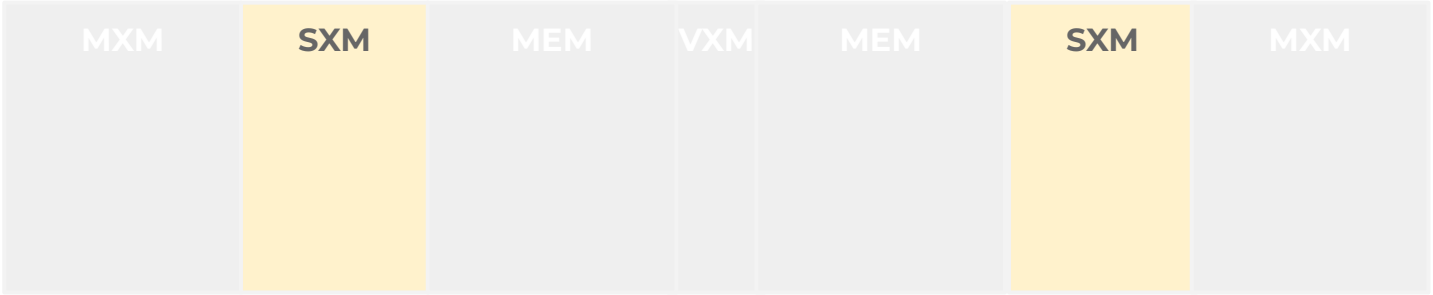
Many concurrent streams are supported in programming model

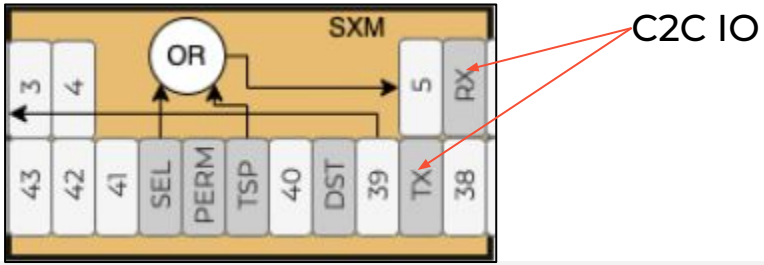
VXM provides a flexible and programmable fabric for Compute

Compute occurs on data locality of passing Stream Tensor

MEM bandwidth supports high concurrency

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 lanes for 20 superlanes

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

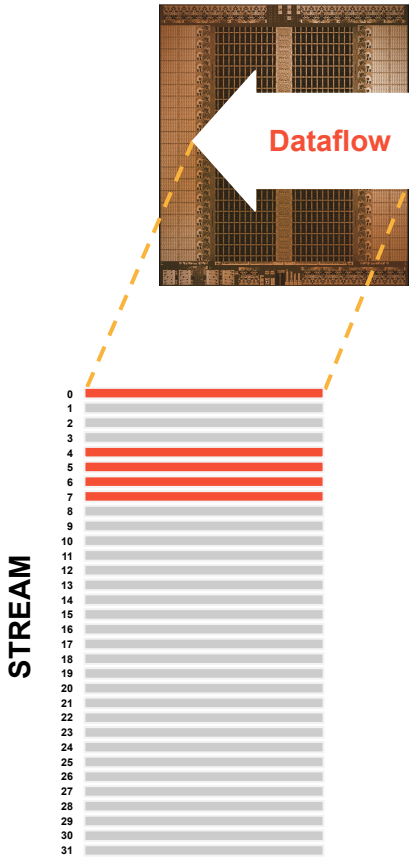
Shift, Rotate, Distribute, Permute, Transpose, Transport to SuperLanes

Streaming Registers

Contrasts to CPU / Register file
- Memory semantics have an address **and** a direction of data flow

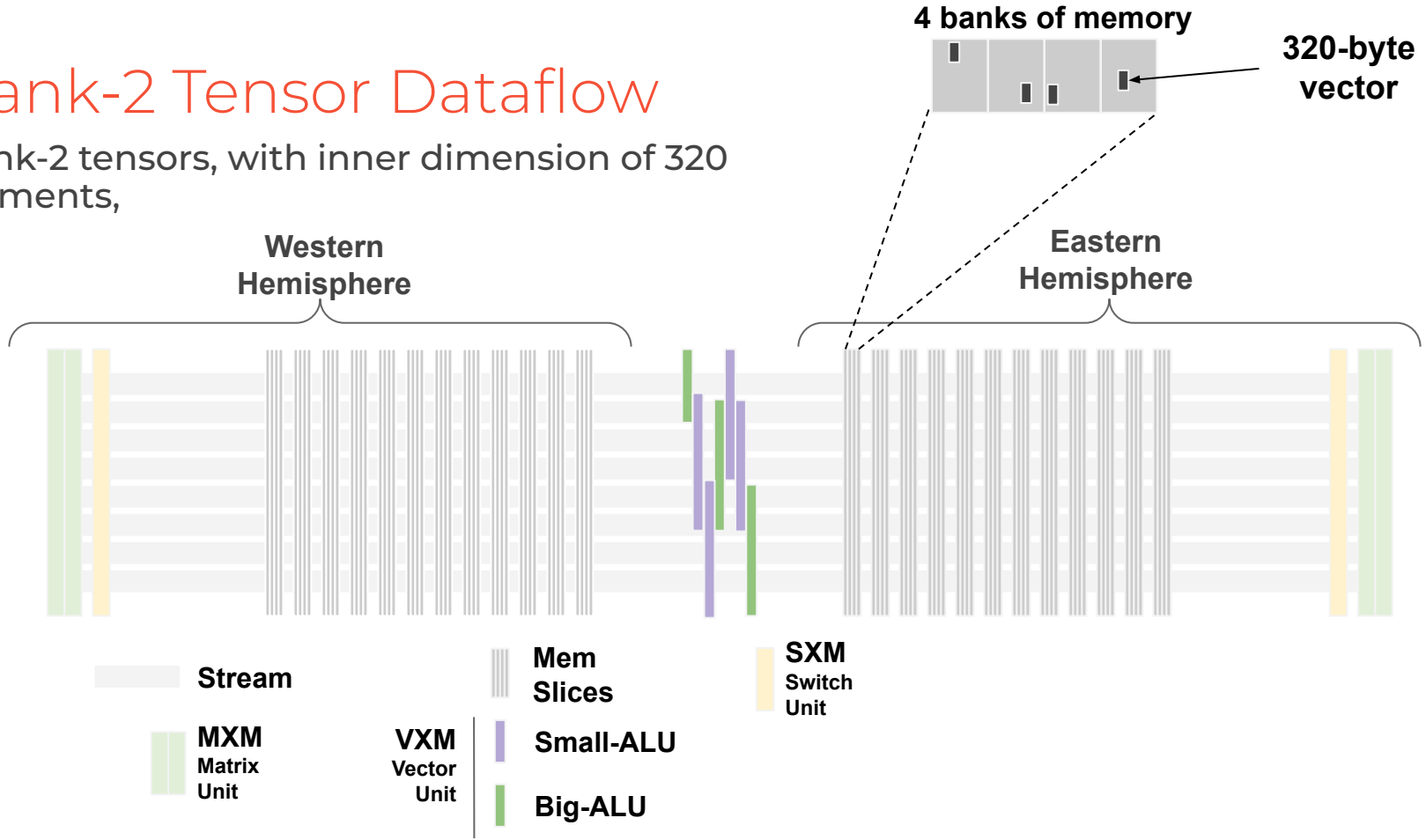
Streaming registers are like a **tensor assembly line** flowing eastward and westward.

Each stream holds a vector of **320 bytes** (i.e. each element in the vector is 1-byte of data)
- **Multiple streams** are used to represent larger multi-byte data types **(eg. FP32)**.



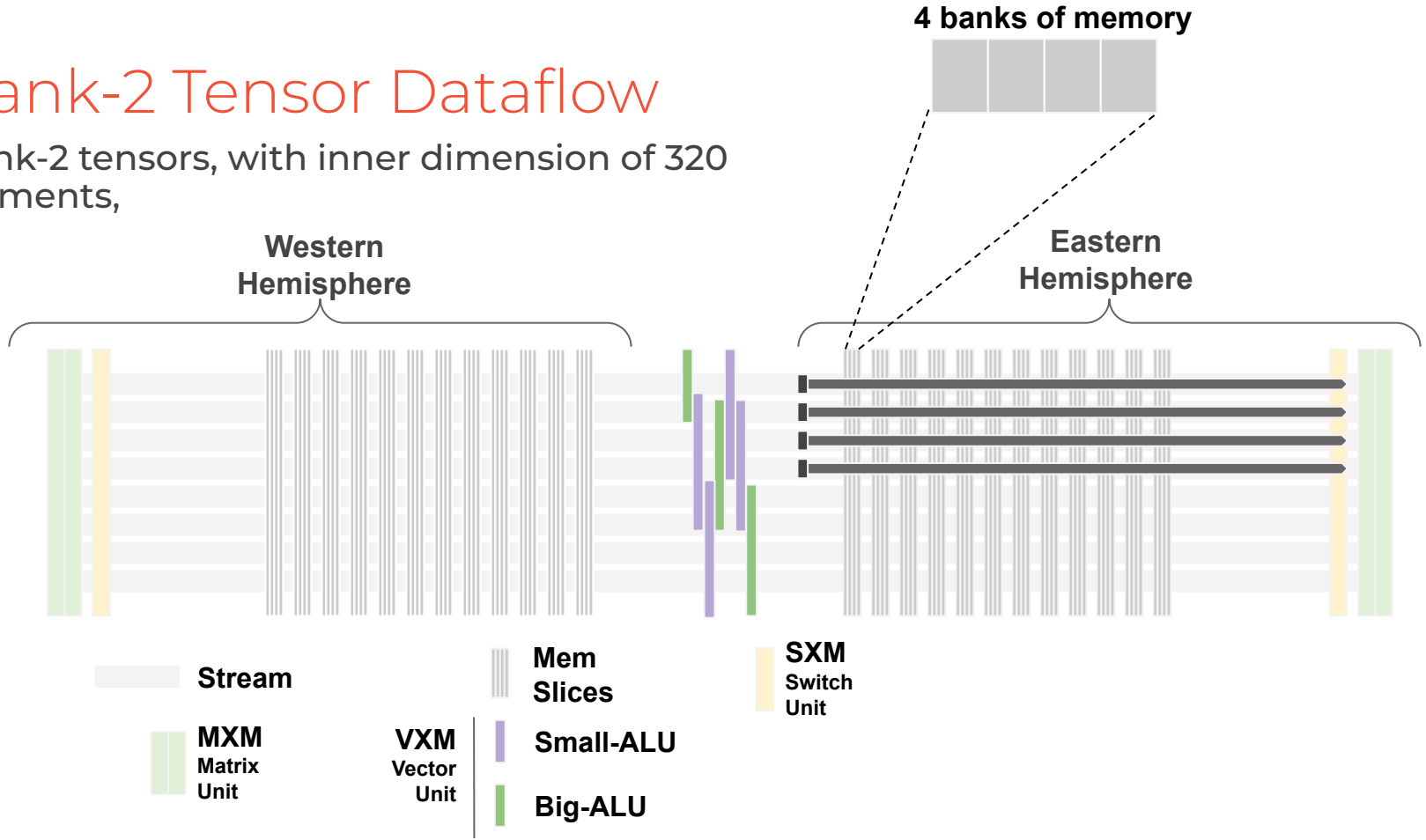
Rank-2 Tensor Dataflow

Rank-2 tensors, with inner dimension of 320 elements,



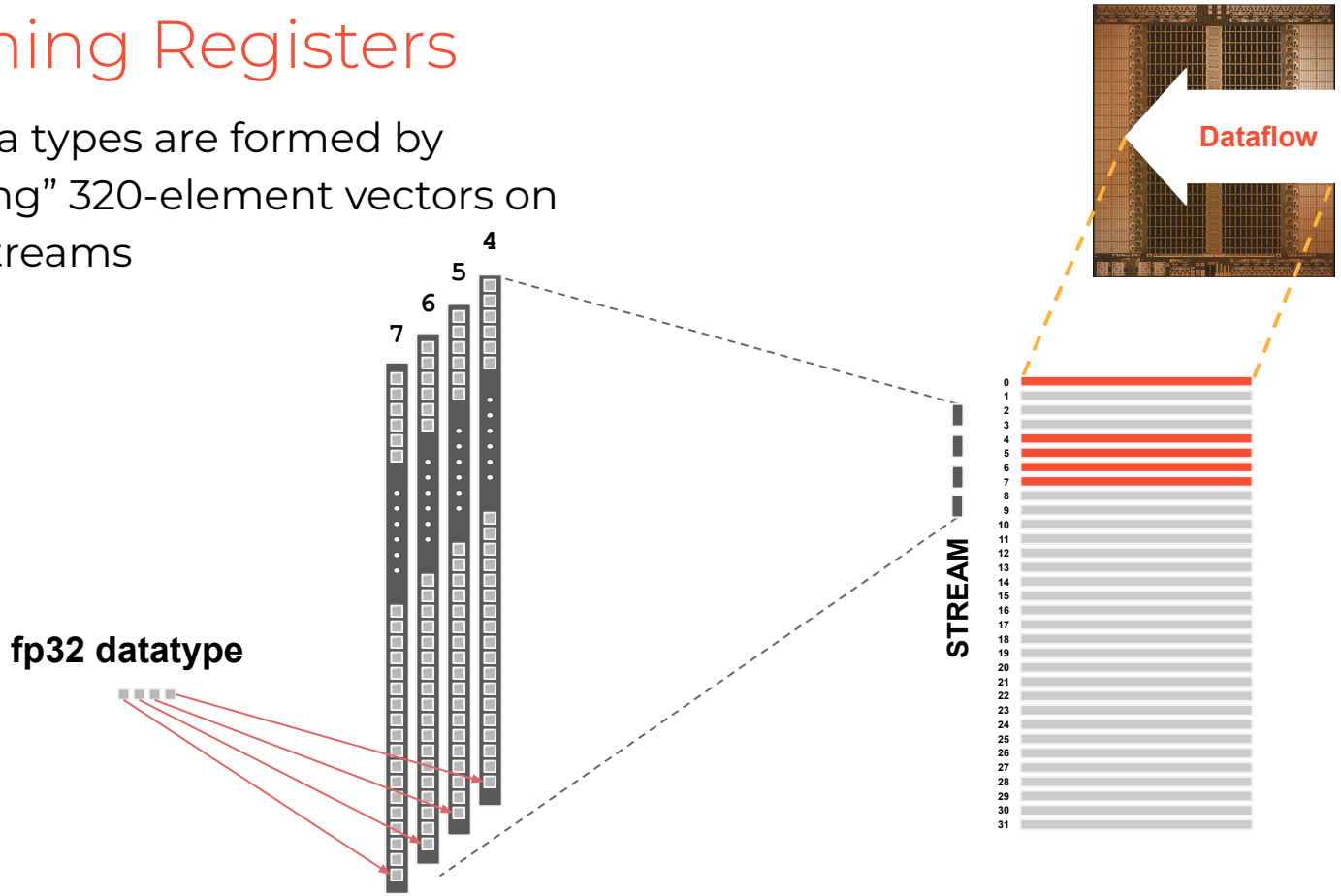
Rank-2 Tensor Dataflow

Rank-2 tensors, with inner dimension of 320 elements,



Streaming Registers

Larger data types are formed by “interleaving” 320-element vectors on multiple streams



An ISA That Empowers Software

Explicit time and space of instruction execution exposed by ISA to the compiler

Each Functional Unit (FU) type provides its own low-level instruction set

Number of FUs of each type and relative positions on chip are exposed to software

Compiler can choose to leverage multiple FUs for more concurrency or more pipelining

Instruction Set

Low level

320-vector ops

Explicit resource selection

Explicit scheduling

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`

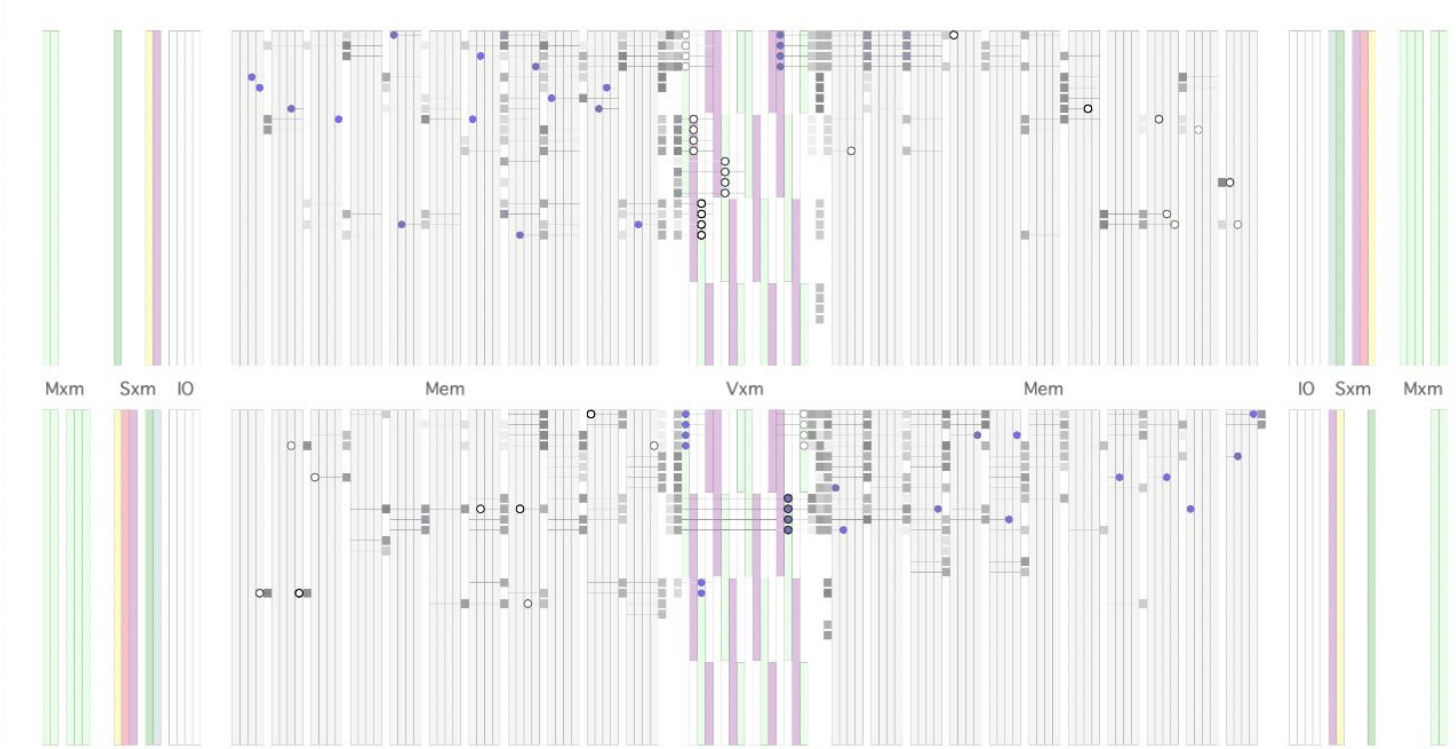
MXM	`LW`
	`IW`
	`ABC`
	`ACC`

SXM	`Shift up/down N`
	`Permute map`
	`Distribute map`
	`Rotate stream`
	`Transpose sg16`

C2C	`Deskew`
	`Send`
	`Receive`

Power of Data Orchestration

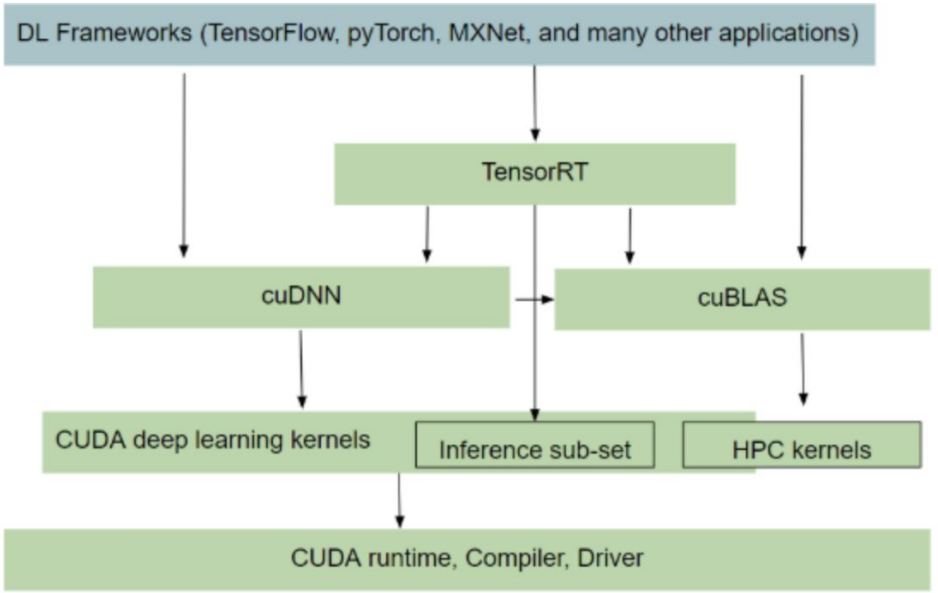
Given to Groq Compiler



Compilation

Traditional HPC Compiler Flow

A: The following graphic shows how cuDNN relates to other software in the stack.
Figure 11. Software stack with cuDNN.



NVIDIA CUDNN DOCUMENTATION

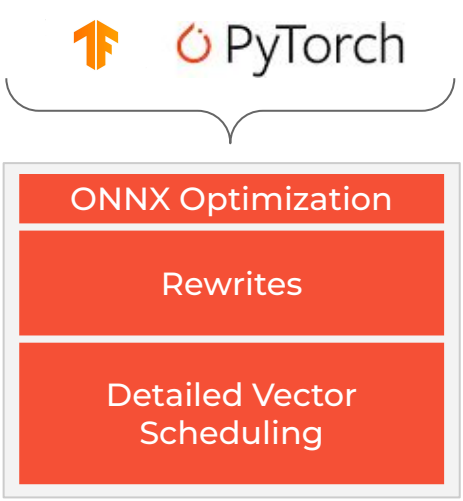
https://docs.nvidia.com/deeplearning/cudnn/developer-guide/index.html

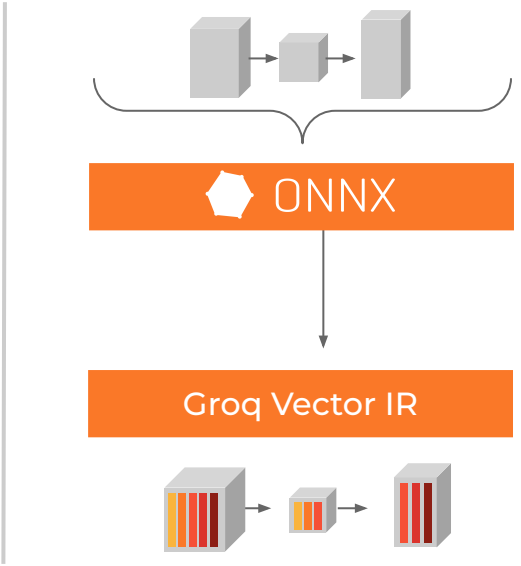
"The most dangerous phrase in the language is "We've always done it this way."

REAR ADMIRAL GRACE HOPPER
Pioneering Computer Scientist
1906–1992

Flow

- **Software-defined hardware** relies on several interfaces:
  - Static-Dynamic interface - compile-time versus runtime
  - Hardware-software interface - exposing the architectural visible-state

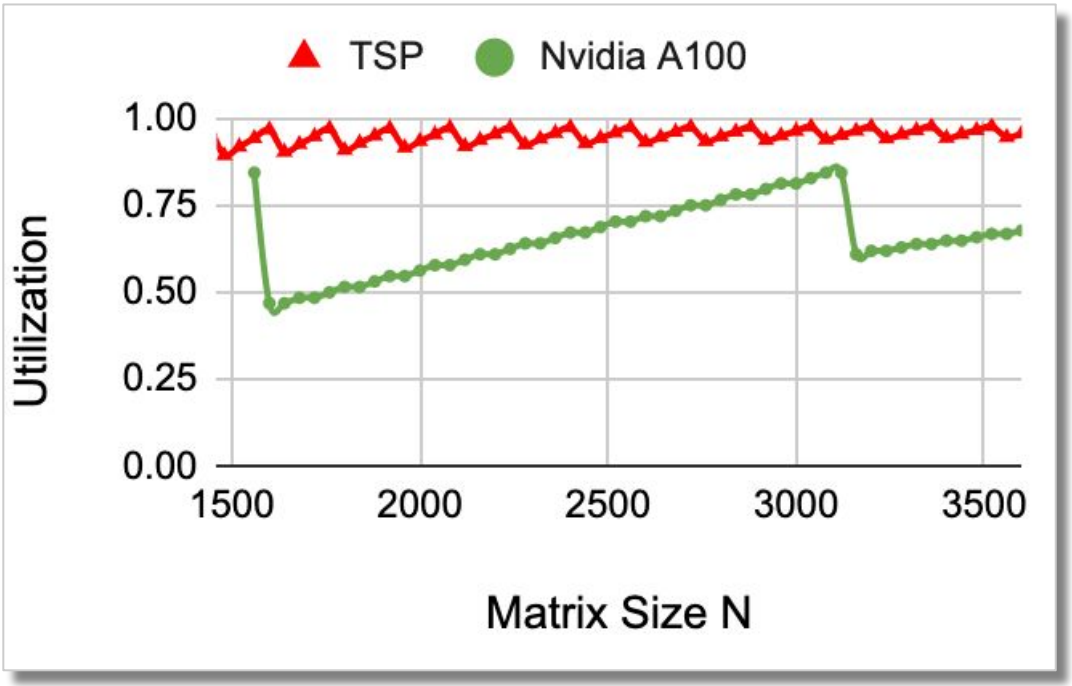




Kernel-less approach to HPC Compilation

GEMM - General Matrix Multiplication

- Vector-Matrix multiplication and Matrix-Matrix multiplication are the workhorse for many ML and HPC workloads
- On-chip memory bandwidth determines how quickly we can ramp up ALUs for vector and matrix operations
- Consistent performance across a range of tensor sizes - less “hardware fitting”
- State-of-the-art (SOTA) results across a range of models and applications
  - CNNs
  - RNNs, LSTMs
  - NLP, BERT
  - Dense linear algebra V*M and M*M



GroqFamily

GroqChip™

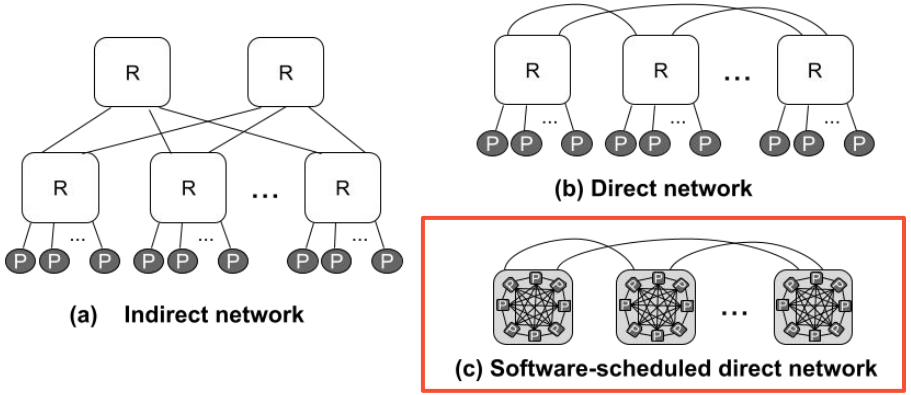
GroqCard™

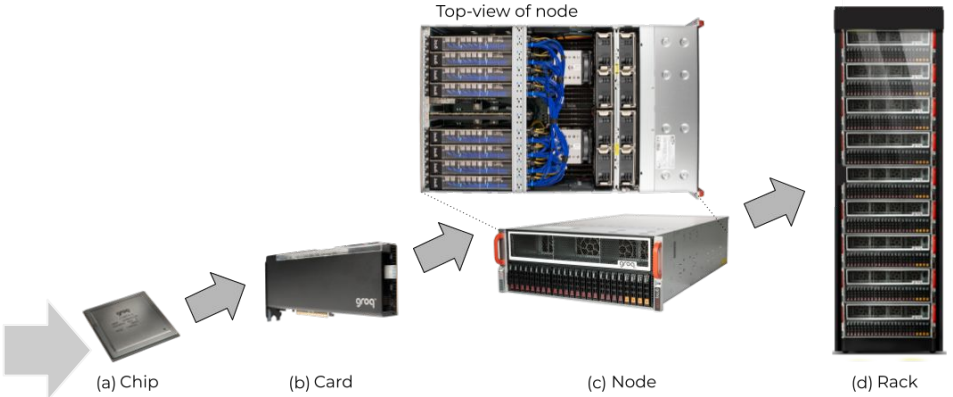
GroqNode™

GroqRack™

Scale-out organization

- Topology objectives
  - Low network diameter
  - Direct network
  - Hierarchical packaging-aware topology
- System packaging hierarchy
- Chip-to-chip (C2C) links and flow control

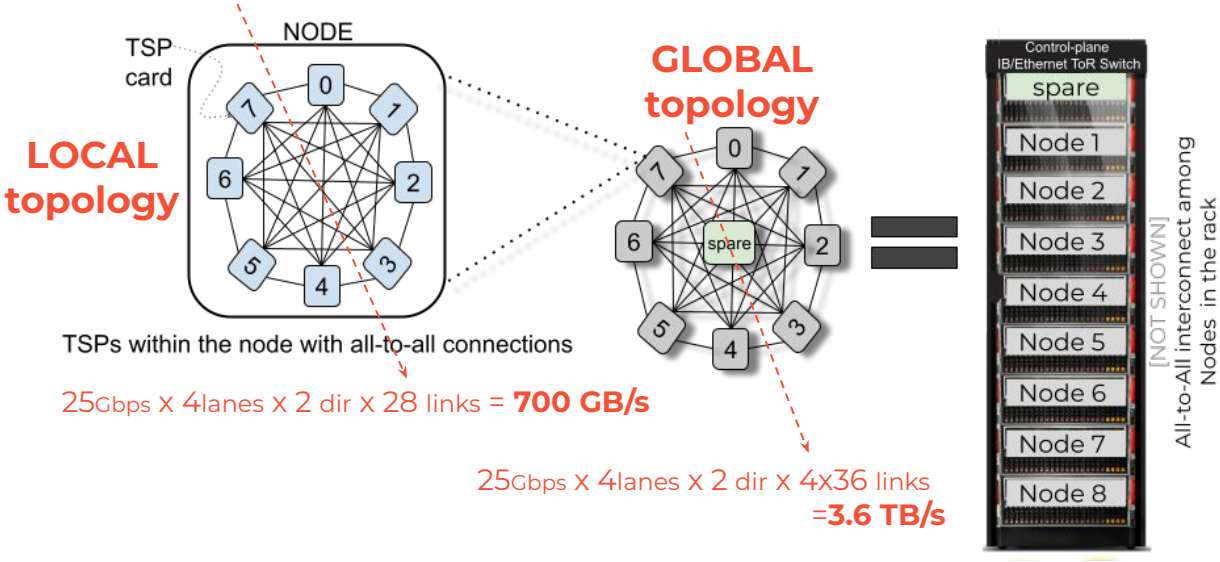




Low-diameter Network

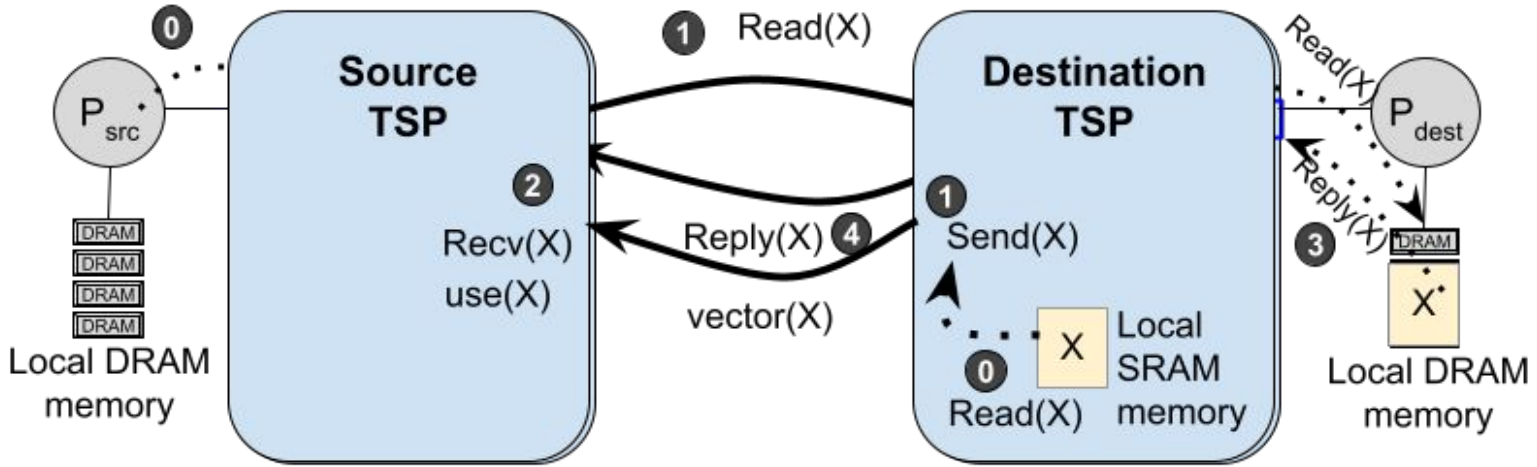
Minimize the number of hops in the network

- The total observed latency and variance increases with the number of hops in the network
- Dragonfly is a *hierarchical* topology minimizes the number of hops taken
  - Local group topology
  - All-to-all global topology
- Exploits packaging locality
- Local topology 2x speedup
- Scalable from 1 TSP to thousands



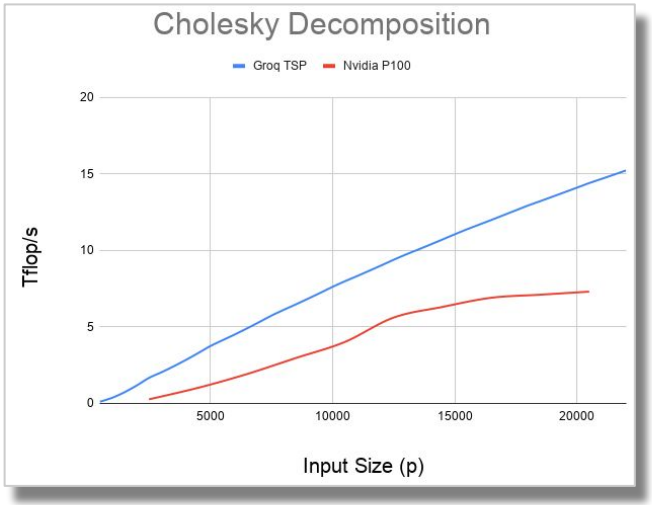
Chip-to-chip (C2C) links and flow control

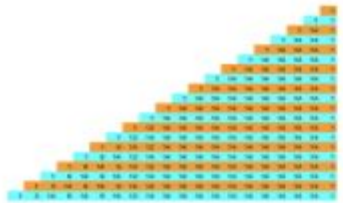
- Simplified communication model provides **logically shared access to global SRAM** which is physically distributed among the TSPs
- C2C links directly connect TSPs
- Comparing conventional RDMA with Groq C2C communication



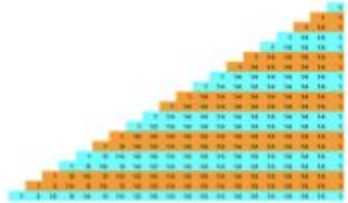
Cholesky Factorization

- Cholesky decomposition is an important technique for solving large-scale system of linear equations.
- Compute is $N^3/3$ for a SPD matrix of size N
- Block-cyclic distributions of 320 rows across chips

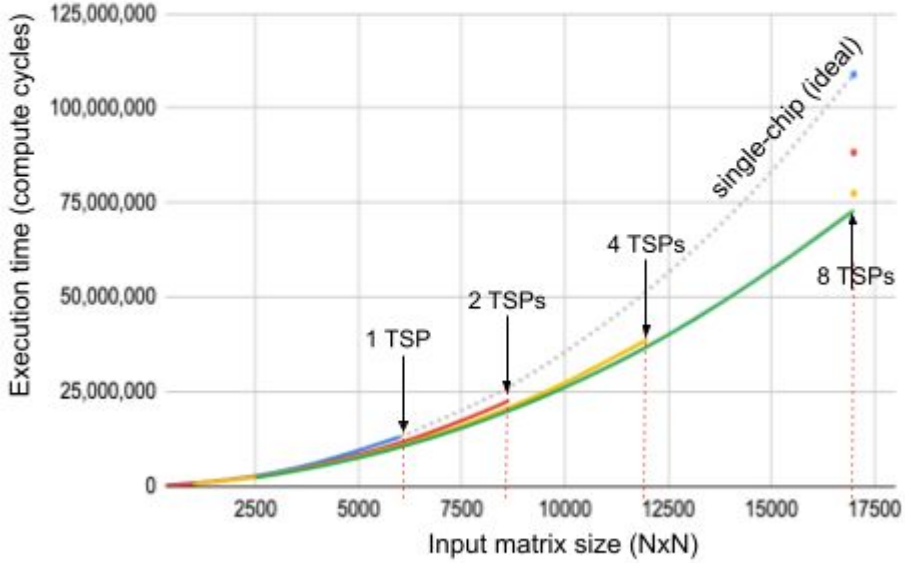




(a) 320 rows interleaved across each chip



(b) blocking across 2 rows and 2-chips



(c) execution time vs problem size (input matrix size) with multiple TSPs

Multichip Compilation

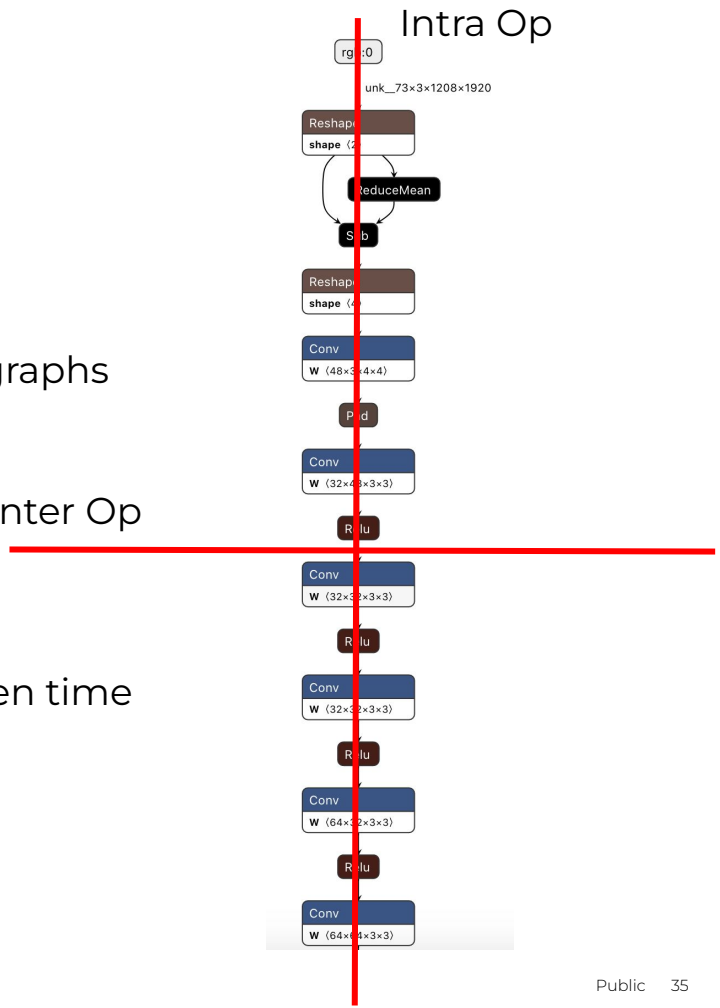
Available soon in next Quarter

Inter Op Partitioning

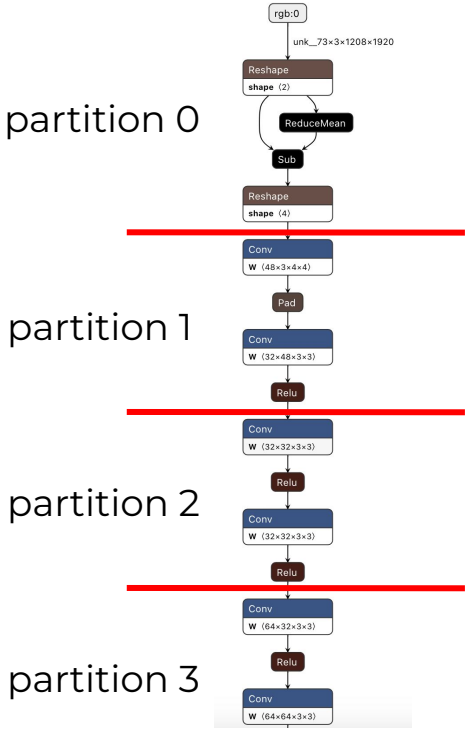
- Model graph split into multiple graphs
- Intermediate activations become inputs to successive graphs
- *Reduces weights on chip*

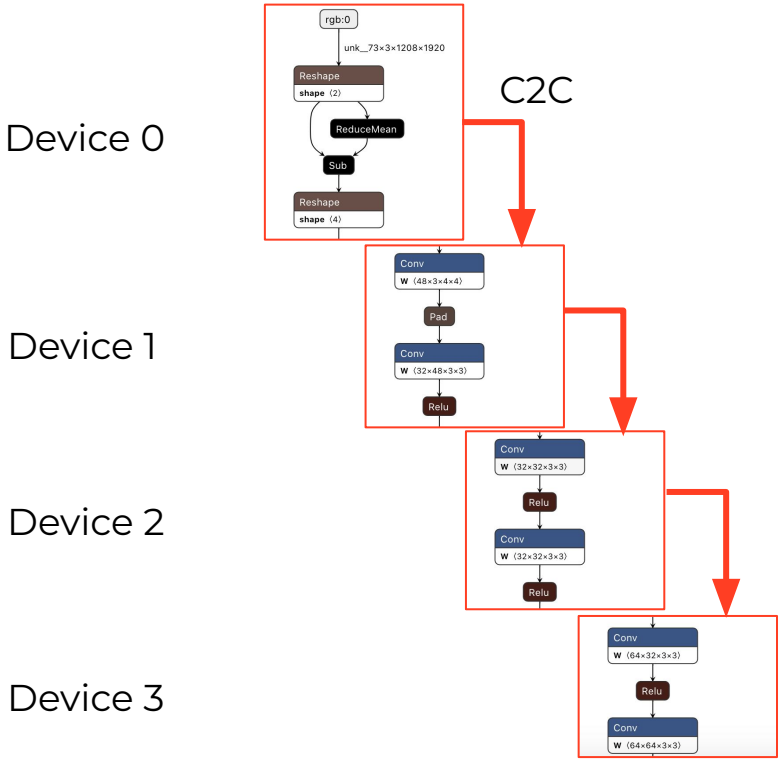
Intra Op Partitioning

- Tile the face of the input
- Operate on a “fractional” portion of the input at any given time
- *Reduces peak activation / input size on the chip*



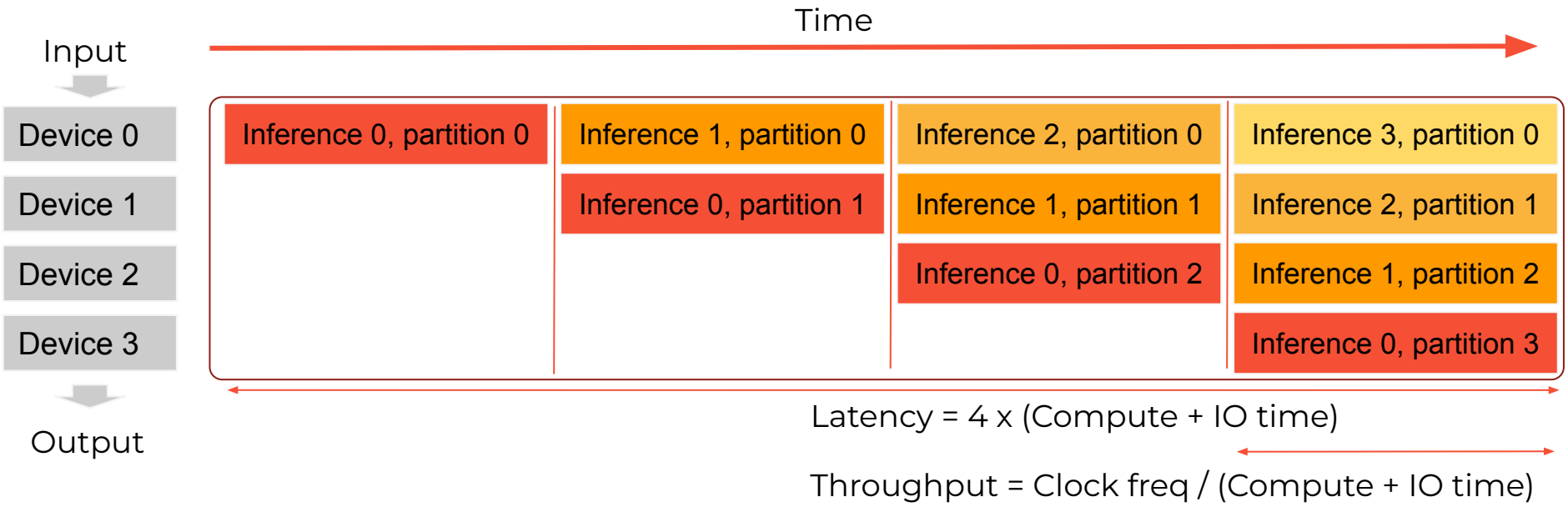
Multichip Inter Op Partitioning





Multichip Pipelining

- Invocation is composed of compute on each of the devices followed by C2C to next subgraph's device
  - Pipeline parallel execution

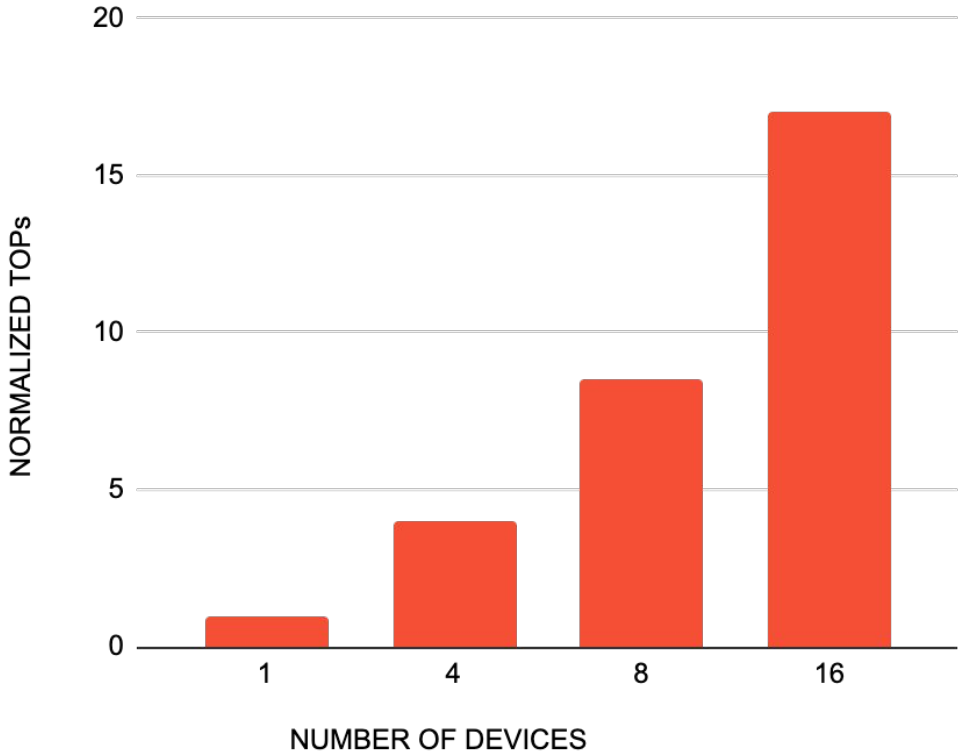


Inter Op Partitioning Results Today

Linear Scaling Achieved

*Scaling **BERT** encoders*

Scaling to 6, 24, 48, and 96 BERT encoders on 1, 4, 8, and 16 TSPs respectively



Results

Natural Language Processing with BERT

Seq Len=128

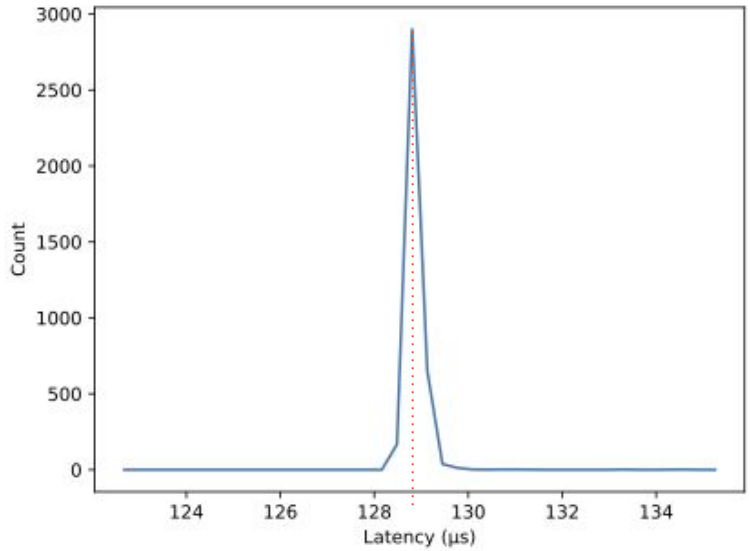
- Spread the layers across TSPs in each node for multi-TSP model parallelism
- Mini-batch (data parallelism) across nodes for throughput
- Goal: Accelerate BERT minimizing **latency** and maximizing **throughput**
- Minimize **latency variance** to provide predictable throughput

T4, CURRENT SOTA (A100) [21] AND THIS WORK BERT-BASE LATENCY (128 SEQUENCE LENGTH).

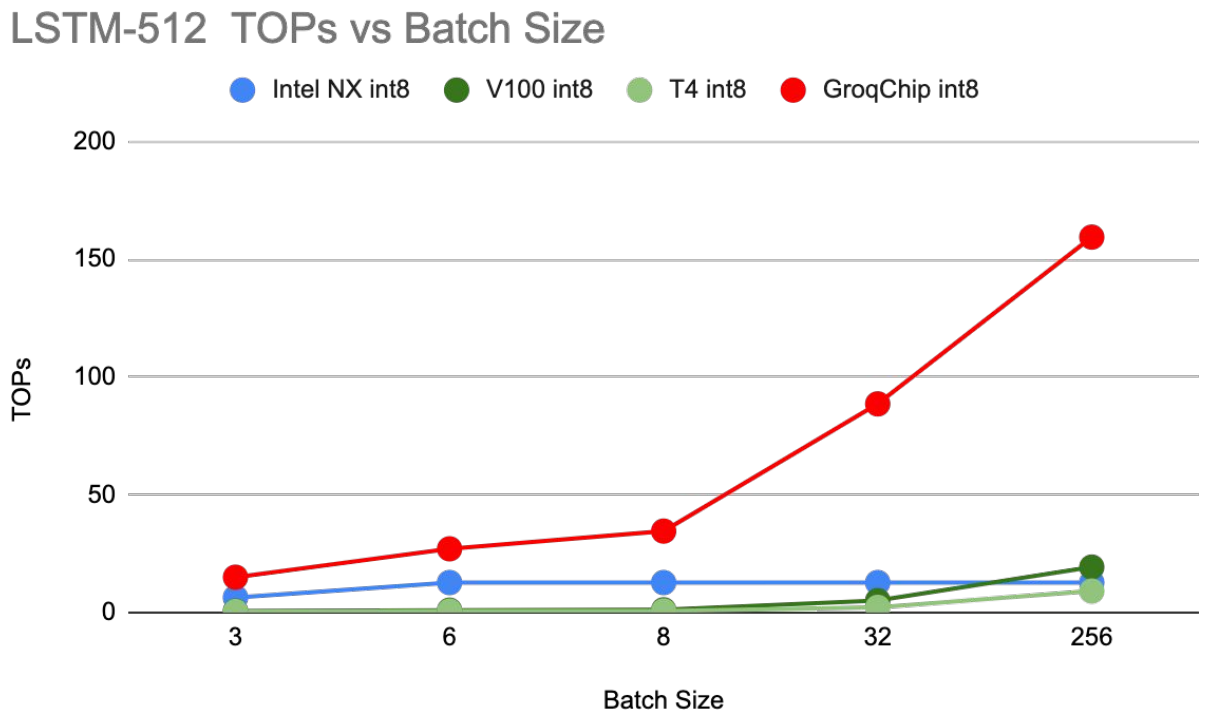
	T4 (µs)	Current SOTA A100 (µs)	This work (µs)	Speedup (This work vs SOTA)
Average	1330	630	128.9	4.8×
95th Percentile	1550	780	129.1	6×
99th Percentile	1570	790	129.5	6.1×

A100 [22], T4 [23] AND TSP [13], [24] SPECIFICATIONS.

Chip	Die Area (mm^2)	Tech Process (nm)	Transistor count (B)	TDP (W)
NVIDIA T4	545	12	13.6	70
NVIDIA A100	826	7	54.2	400
Groq TSP	725	14	26.8	275



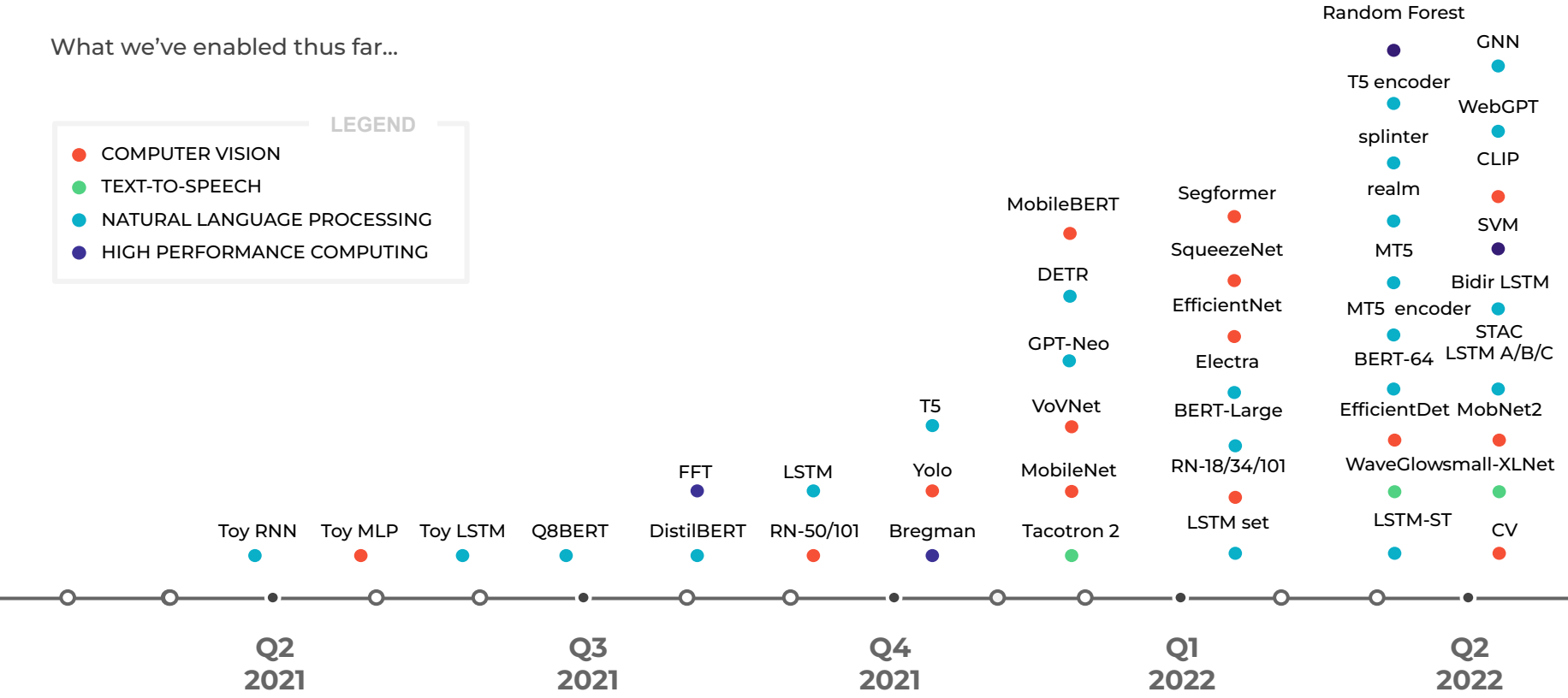
Achieving SOTA Results



Nvidia results from publicly available data on nvidia.com
LSTM results from Intel sponsored paper, FPT 2020, available on intel.com

Groq Exponentially Supporting More Workloads

What we've enabled thus far...



Summary and Takeaways

- Low-latency and high-throughput are necessary…
- Delivering **predictable** and **repeatable** performance is critical for many user-facing applications
  - Batch-1 inference is important for responsiveness and delivering quality-of-service (QoS) that is impossible to do with more traditional microarchitectures using crossbars, cache hierarchies, etc.

- **Determinism** enables **software-defined hardware** and entails a design philosophy that spans both hardware and software
  - ISA is not about abstraction of hardware details, but about **exerting control of underlying hardware**
    - 144 independent instruction control units (ICUs) of the TSP
  - Expose the architecturally-visible state (GPRs, SRAM, instruction buffers, etc)
  - Software-based replay and exception handling

- Extending the single-chip TSP determinism to the multiprocessor using **software scheduled networking** to explicitly schedule tensors on the network links

- **Synchronous communication** model allows for large-scale machine learning

groq™

We're Hiring!

Join us...