Challenges and opportunities in ML accelerator architectures

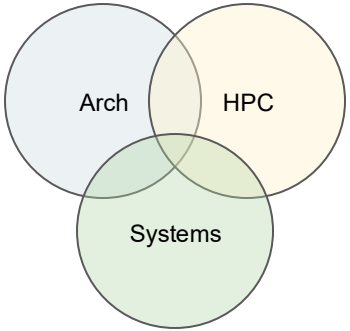
Kiran Ranganath
SambaNova Systems

Bio

Principal Researcher, SambaNova Systems

PhD, UC Riverside
Advisor: Dr. Daniel Wong

Research interests: Computer architecture, high-performance computing, Systems for ML



Introduction

Dawn of transformer-based LLMs

- OpenAI - GPT, GPT 2 (1.5B), GPT 3 (13B), GPT 4 etc.
- Meta - Llama, Llama 2 (7B, 13B, 70B), Llama 3.x (8B, 70B, 405B), Llama 4 (100B, 400B)
- Mistral - 3B, 7B, 8B, 8x22B etc.
- BigScience - Bloom (176B)
- DeepSeek (671B)
- *Many more..*

We need the ability to ***train*** and ***serve*** models at scale.
Train = Pre-train + Fine-tune
Serve = Inference

Transformers for NLP

[NeurIPS'17] **Attention is all you need**[1]

Key takeaways for HPCA enthusiasts:

- ***Fast*** to train
- Support for ***longer sequence*** lengths/contexts
- It converted NLP into a **Highly parallelizable** compute problem

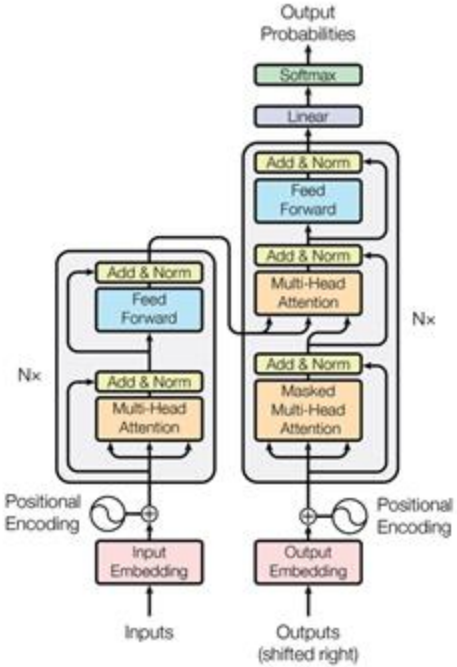


Figure 1: The Transformer - model architecture.

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

Breaking down the transformer model architecture

DeepSeek v3
N = 61
h = 128

- *Nx* decoders
- Each decoder
  - Multi-head attention (MHA) (*hx* heads)
    - Scaled Dot Product Attention (SDPA)
    - Q, K, V Linear
- Complexity?
  - Bulk : `N x h x (SDPA + Q + K + V + Feed forward)`
  - GeMM (Linear & MatMul) throughput dominates model performance.

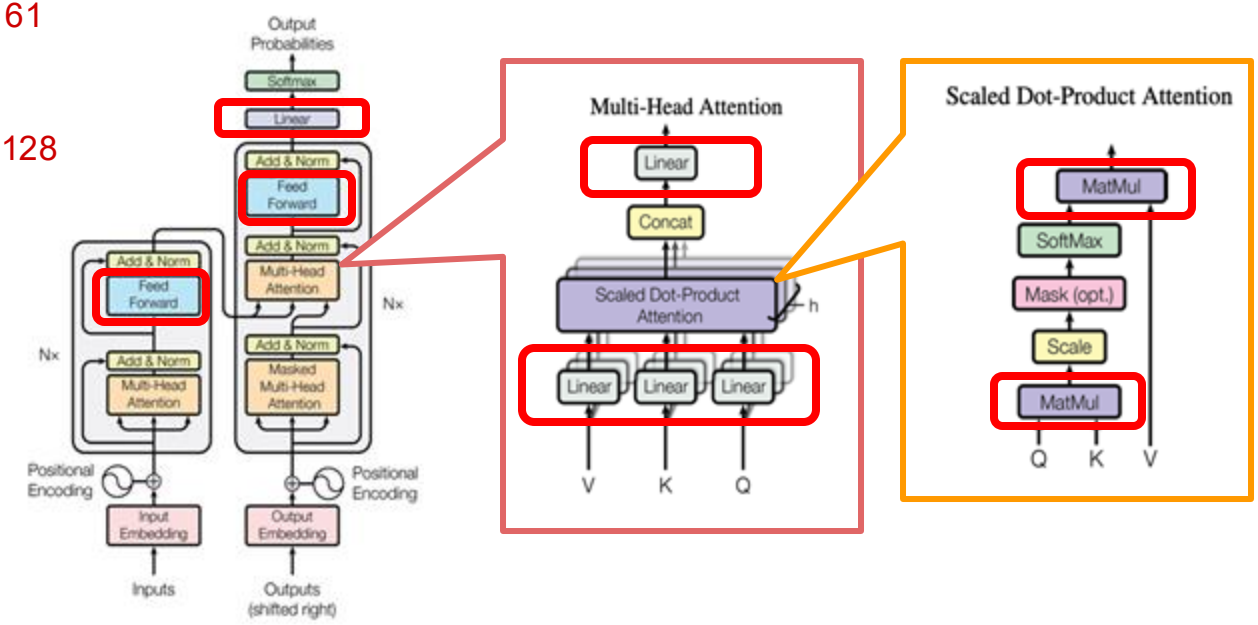


Figure 1: The Transformer - model architecture.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin.2017. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 6000–6010.

Challenge 1: ML as an HPC problem

Speeding up General Matrix Multiplication (GeMMs)

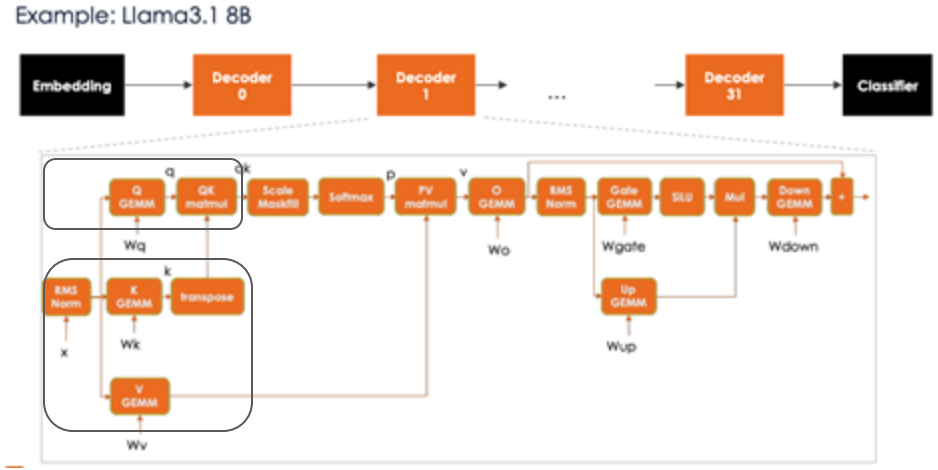
- CPU optimizations?
  - Effective use of caches, locality, etc.
  - Libraries - OpenBLAS etc.
  - Not suitable for large matrices
  - Can extract throughputs of <1 TFLOPs per chip
- GPU optimizations?
  - Effective use of global memory, shared memory, high-bandwidth memory (HBM), tiling, etc.
  - Proprietary libraries like CuBLAS offer near to linear scaling.
  - GeMMs can be scaled across multiple GPUs
  - Throughputs can range from 10+ TFLOPs per chip

```c
#include <stdio.h>

// Function to perform GeMM: C = alpha * A * B + beta * C
void gemm(int M, int N, int K,
          float alpha, float *A, float *B,
          float beta, float *C)
{
    for (int i = 0; i < M; ++i) {
        for (int j = 0; j < N; ++j) {
            float sum = 0.0f;
            for (int k = 0; k < K; ++k) {
                sum += A[i*K + k] * B[k*N + j];
            }
            C[i*N + j] = alpha * sum + beta * C[i*N + j];
        }
    }
}
```

Breaking down Llama 3.1 8B inference

- 32 decoders
- 9 GeMMs per encoder
- Can support 128k sequence length
- Model parameters = ~16 GB (BF16)
- KV cache size = up to 4GB / 1k sequence / batch
- For sequence length = 16k,
  Memory required = ~1TB
  Model FLOPs = ~8.8 TFLOPs



Need of the hour

- **Faster & larger on-device memory**
  - High-bandwidth memory (HBM), SRAM etc.
- **Higher TFLOPs**
  - Higher core frequency
  - Pack more compute threads

Challenge 2: Thinking beyond GPUs

Breaking down Llama 3.1 8B inference

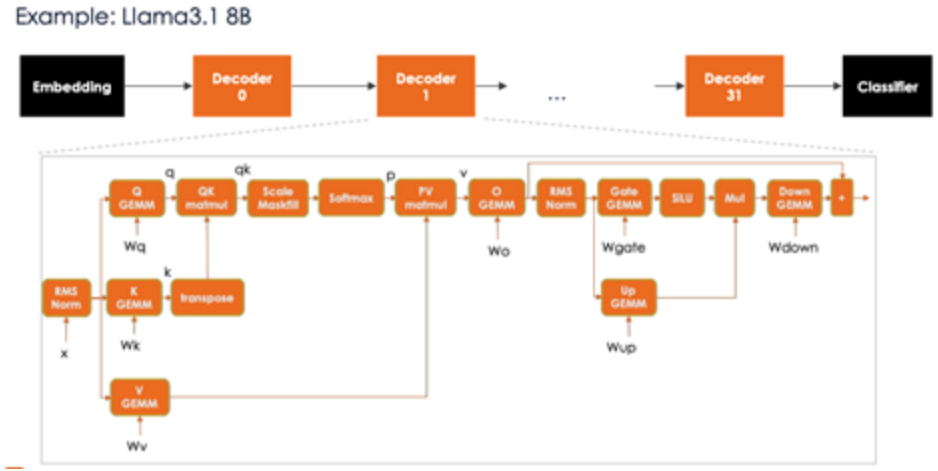
Llama 3.1 8B config:

Memory required = 1TB

Model FLOPs = ~8.8 TFLOPs

How to execute this graph?

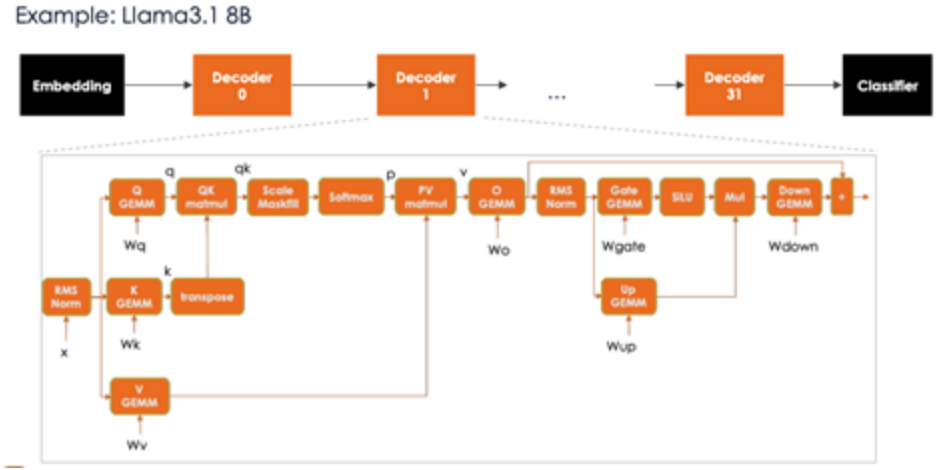
- Block by block?
- Section by section?



Block by block

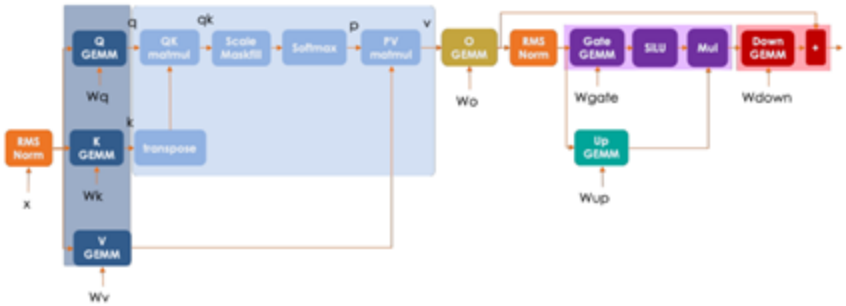
Execute each of the blocks individually one-by-one (kernel-by-kernel) - GPU-style

- Native CUDA support in PyTorch
- Leverage CuBLAS, CuDNN, etc.
- Performance tuning can be messy
  - Kernel fusion



Section-by-section

- These models can be executed as ***DataFlow*** graphs.
- Break down the model into sections
  - Fewer kernel calls
- Generate output tokens as the data flows through the ***Directed Acyclic Graph (DAG)***
  - Model FLOP utilization (MFU) ⬆
  - Model Bandwidth utilization (MBU) ⬆
  - Total Cost of ownership ⬇



[1] R. Prabhakar, "SambaNova SN40L RDU: Breaking the Barrier of Trillion+ Parameter Scale Gen AI Computing," 2024 IEEE Hot Chips 36 Symposium (HCS), Stanford, CA, USA, 2024, pp. 1-24, doi: 10.1109/HCS61935.2024.10664717.

Need of the hour

- ***Reconfigurable ML accelerators***
  - Field Programmable Gate Array (FPGA)
    - Programmable at *fine-grain* granularity (Lookup tables, Flip-flops, etc.) - ***very flexible***
    - *Programming is hard!* 😓
  - Coarse Grain Reconfigurable Architecture (CGRA)
    - Programmable at *coarse-grain* granularity (ALUs, etc.)
      - ***flexible*** enough to lay out transformer DAG on chip
    - *Programming is not so hard!*

Rise of DataFlow-based ML accelerators

- SambaNova's SN40L Reconfigurable DataFlow Unit (RDU)
  - 1,040 compute cores
  - 3-tier memory within a chip
    - 512 MB of programmable memory (SRAM) - several PB/s
    - 64 GB of HBM - 25+ TB/s
    - 1.5TB of DDR (device memory) - 1600 GB/s
- Groq's Language Processing Units (LPU)
  - Large SRAM ~250 MB per chip
- Cerebras Wafer-scale engine (WSE-3)
  - 900k compute cores
  - 44 GB of SRAM

SambaNova SN40L

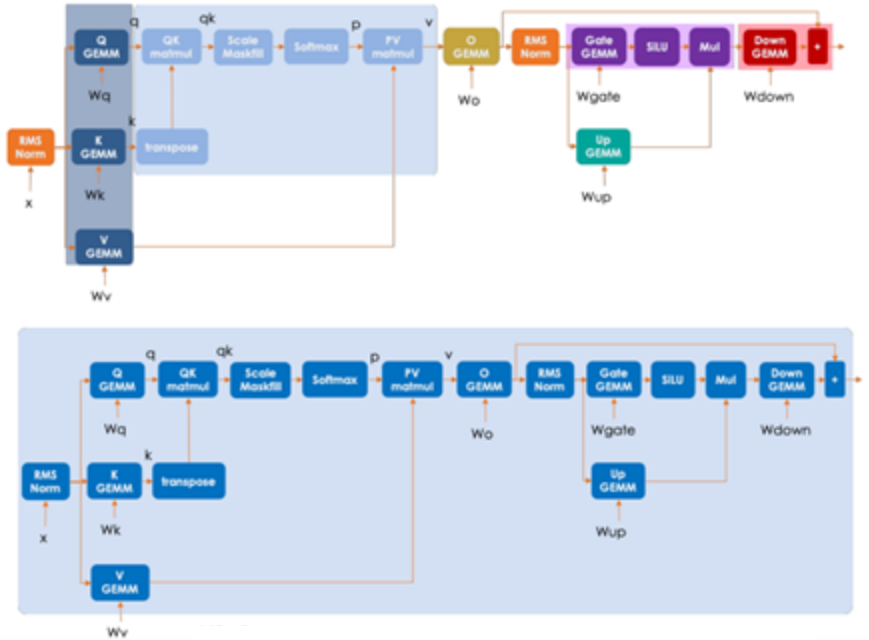
Cerebras WSE-3

Challenge 3: Extracting the full throughput

Problems with DataFlow executions

- **High** *Kernel launch overheads*
  - Reconfigure after each section
- **Low** *Data Locality*
  - After section is executed
    - Drain data & arguments out of chip
  - Before start of next section
    - Load data & arguments back to chip

Ideal scenario - *Execute as a single section*



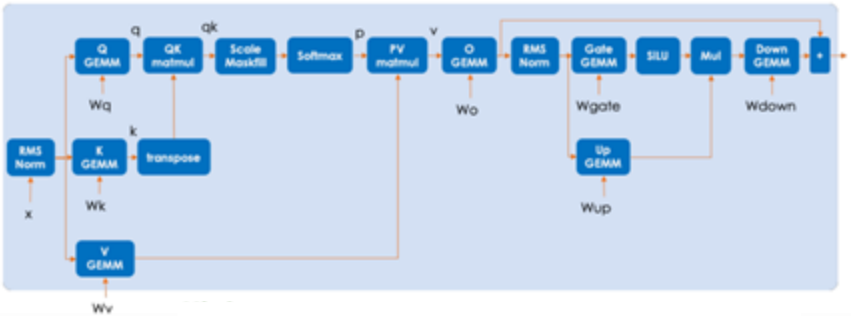
[1] R. Prabhakar, "SambaNova SN40L RDU: Breaking the Barrier of Trillion+ Parameter Scale Gen AI Computing," 2024 IEEE Hot Chips 36 Symposium (HCS), Stanford, CA, USA, 2024, pp. 1-24, doi: 10.1109/HCS61935.2024.10664717.

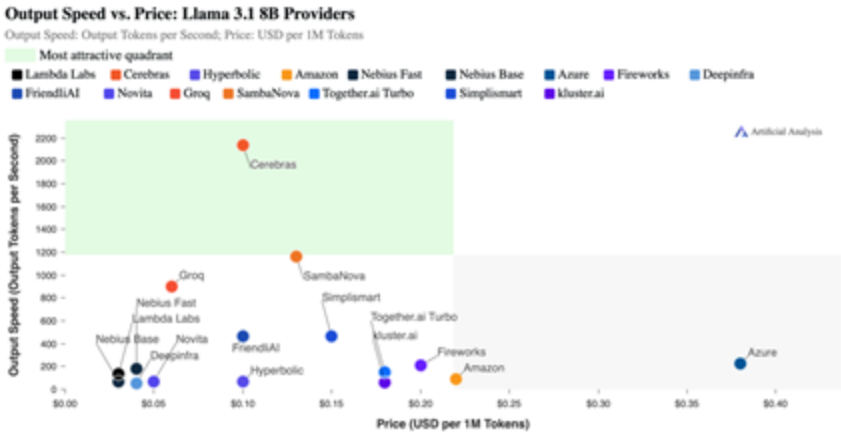
Inference as DataFlow

- Single kernel launch overhead
- Exploit data locality
- (Recap) Llama 3.1 8B inference

  Memory required = 1TB

  Model FLOPs = ~8.8 TFLOPs
  - Can it fit on a single device?
    - Mostly no!
    - Group devices to work in tandem.
- Roofline of single section execution
  - 1000+ tokens/s output speed🔥





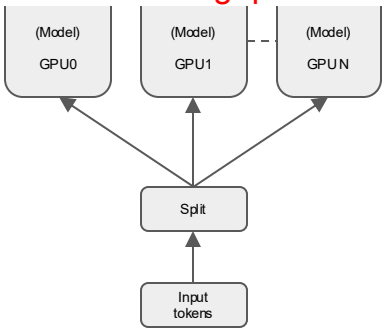
Source: Artificial analysis

Challenge 4: Scalability

Data Parallel

- Multiple *replicas* of the model across devices
  - Process different inputs (e.g., batches of tokens or images)
  - Must fit in a single device

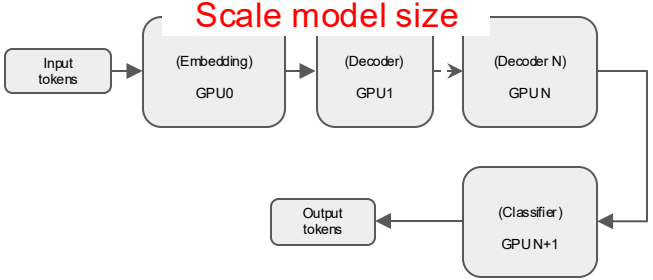
Scale throughput



Model Parallel

- *Split* the model across devices
  - Pipeline bubbles
  - Not generalizable

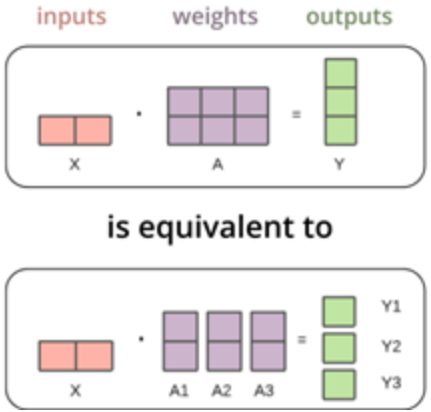
Scale model size



Tensor Parallel

- Multiple *replicas* of the same operation processing *different tensors* in parallel
  - Sensitive to communication latency & bandwidth

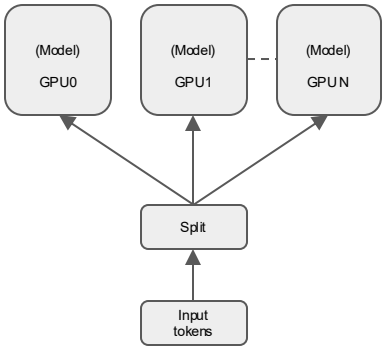
Scale memory & compute utilization



Importance of P2P communication

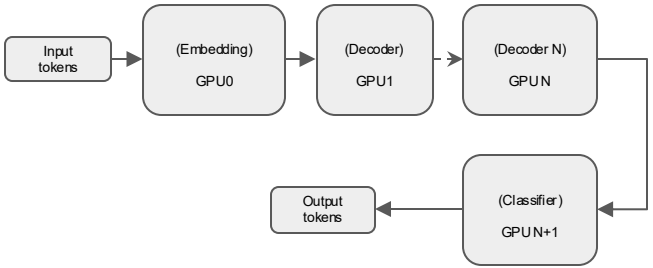
Data Parallel

- Gradient sync



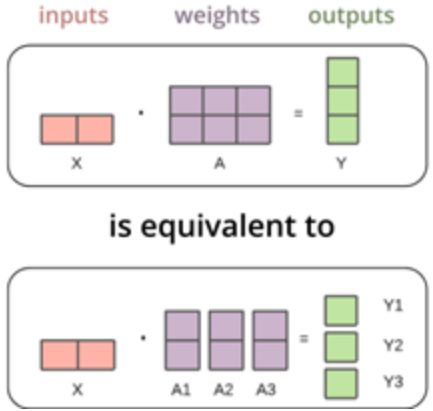
Model Parallel

- P2P send and receive

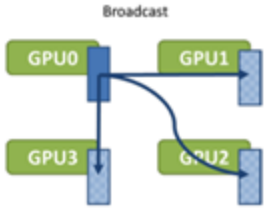


Tensor Parallel

- Many types of sync
- Collective communication calls

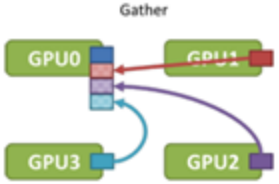


Collective communication calls & scenarios

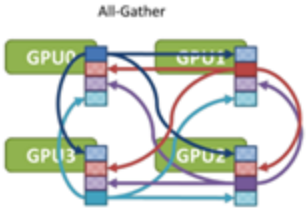


DP/TP
- Replicate Model params/ intermediate results/shared input across devices

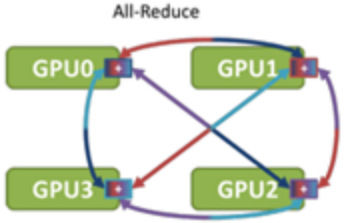
TP
- Distribute intermediate checkpoints/activations across devices



TP
- Gather outputs after TP op for consecutive single-chip ops (like Norm etc.)

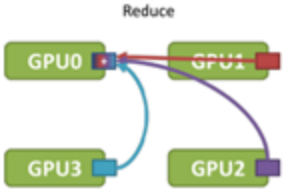


TP
- Gather outputs after TP op like Embedding.



DP
- Gradient sync

TP
- Result of 2d-sharded GeMMs



TP
- Cross Entropy

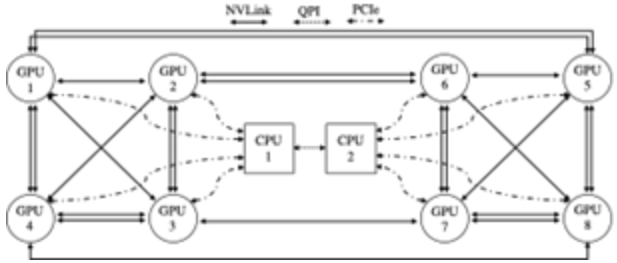
Need of the hour

- **High-bandwidth communication (p2p)**
  - PCIe
  - NVLink
  - CXL etc.
- **HW/SW co-design**
  - Runtime CCL optimization

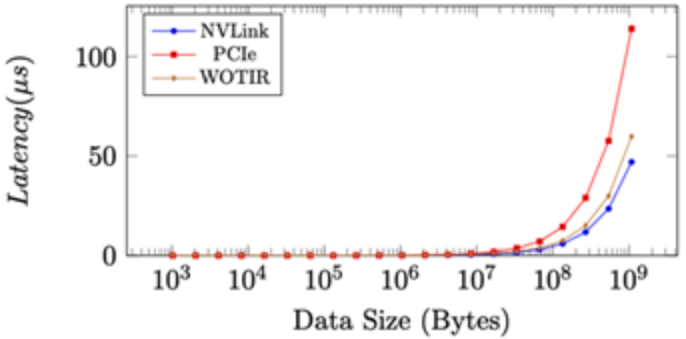
Challenge 5: Designing Low-power systems

System topology

- 8x device cube mesh topology is the common deployment
- Specialized P2P links
  - Nvidia NVlink
  - AMD Infinity fabric
  - CXL
  - PCIe
- Each device can support up to 6 links
- Scaling CCL without A2A connectivity is crucial[1]
  - Low-power
  - Low TCO



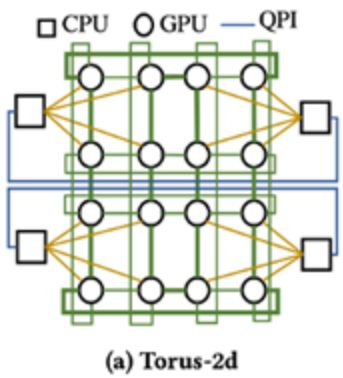
Cube-mesh topology

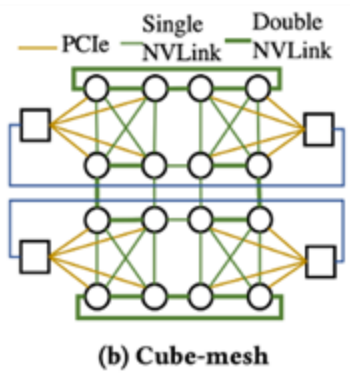


[1] K. Ranganath, A. Abdolrashidi, S. L. Song and D. Wong, "Speeding up Collective Communications Through Inter-GPU Re-Routing," in IEEE Computer Architecture Letters, vol. 18, no. 2, pp. 128-131, 1 July-Dec. 2019, doi: 10.1109/LCA.2019.2933842. keywords: {Graphics processing units;Bandwidth;Routing;Machine learning;Servers;Training data;Interference;Collective communication;GPU;interconnect},

Scaling systems beyond 8 devices

- Specialized interconnects
  - Nvidia NVswitch
    - A2A connectivity up to 16 devices/node
  - High power consumption
  - Low TCO
- Efficient scaling with PCIe
  - Torus
  - Cube-mesh
  - Low-energy
  - Near to linear scaling[1][2]
  - Higher TCO





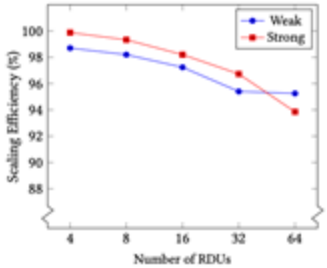


Figure 10: Scaling efficiency on SN30 and RDARuntime

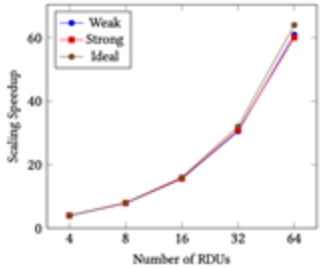


Figure 11: Scaling speeedup on SN30 and RDARuntime

[1] Kiran Ranganath, Joshua D. Suetterlein, Joseph B. Manzano, Shuaiwen Leon Song, and Daniel Wong. 2021. MAPA: multi-accelerator pattern allocation policy for multi-tenant GPU servers. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '21). Association for Computing Machinery, New York, NY, USA, Article 99, 1–14. https://doi.org/10.1145/3458817.3480853
[2] Benjamin Glick, Arjun Sabnis, Renate Kempf, Arnav Goel, Aarti Lalwani, Guoyao Feng, and Kiran Ranganath. 2023. RDARuntime: An OS for AI Accelerators. In Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W '23). Association for Computing Machinery, New York, NY, USA, 1576–1587. https://doi.org/10.1145/3624062.3624235

Challenge 6: DeepSeek, Llama 4 and next?

Mixture of experts (MoE)

- The success of DeepSeek and Llama 4 demonstrated the benefits of MoE
- Monolithic models are slow and prone to hallucinations
- Benefits of MoE
  - Specialized experts for tasks, languages, etc.
  - Activate only a “subset” of experts for an input token
  - Scaling up to billions of parameters with fixed compute cost
  - Better accuracy than monolithic models
  - Sparsity

Summary of challenges & HPCA opportunities

Challenges	HPCA search space
1. ML as an HPC problem	Algorithms, roofline analysis
2. Thinking beyond GPUs	Hardware/software architecture
3. Extract the full-throughput	Compiler, roofline analysis, architecture, runtime
4. Scalability	Hardware-software codesign, runtime
5. Design low-power systems	Hardware-software codesign, runtime
6. Mixture of Experts	Algorithms, roofline analysis, Hardware-software codesign, runtime

Thank you!