DBs + GPUs:
Where are we and what's next?

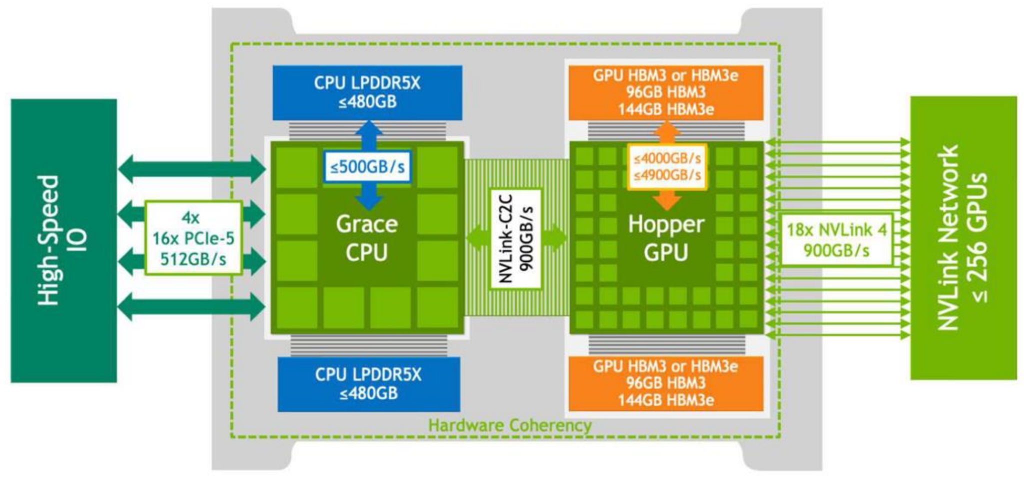
Bowen Wu

Systems Group @ ETHz

Lunch Seminar, 31/10/2025

Why run databases on GPUs?

	Active Threads	Memory Bandwidth	Memory Capacity
Nvidia V100 PCIe	163,840	897 GB/s	16 GB
Nvidia A100 SXM	221,184	1,935 GB/s	80 GB
Nvidia H100 PCIe	233,472	2,039 GB/s	80 GB
Nvidia H200	233,472?	4.8 TB/s	141 GB
AMD MI300X	?	5.3 TB/s	192 GB



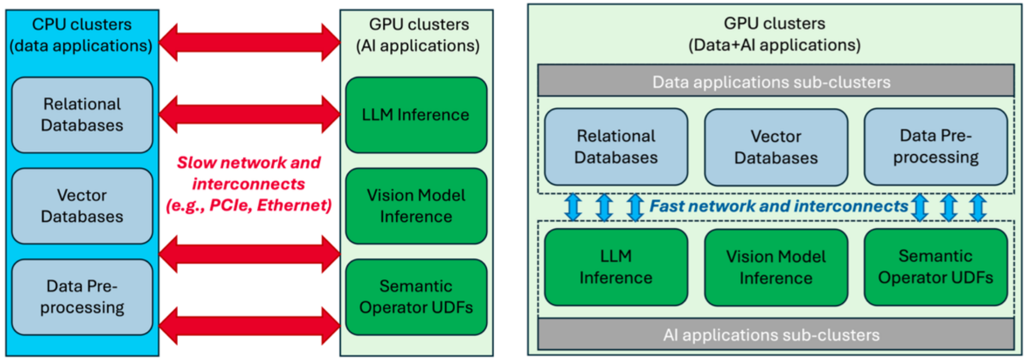
Massive ***parallelism*** and high memory ***bandwidth*** make GPUs suitable for accelerating databases.

Fast ***interconnects*** and ***networks*** make it possible to process ***large*** datasets at an unprecedented speed.

Why run databases on GPUs?

Microsoft inks $33 billion in deals with 'neoclouds' like Nebius, CoreWeave — Nebius deal alone secures 100,000 Nvidia GB300 chips for internal use

News By Sunny Grimm published October 2, 2025



(a) Existing architecture. **(b) Prospective architecture.**

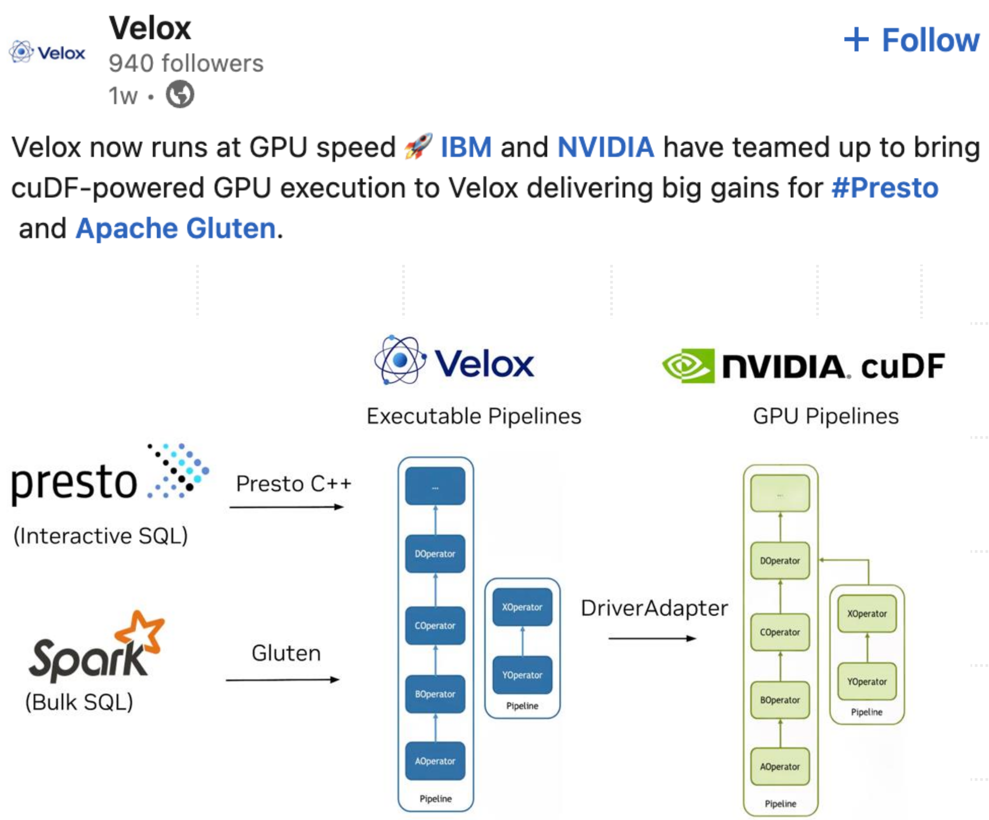
Data centers and clouds are being increasingly dominated by GPUs, which cannot be fully utilized by AI.

Making both DBs and AI run on the GPUs enables optimizations to make both systems run more efficiently.

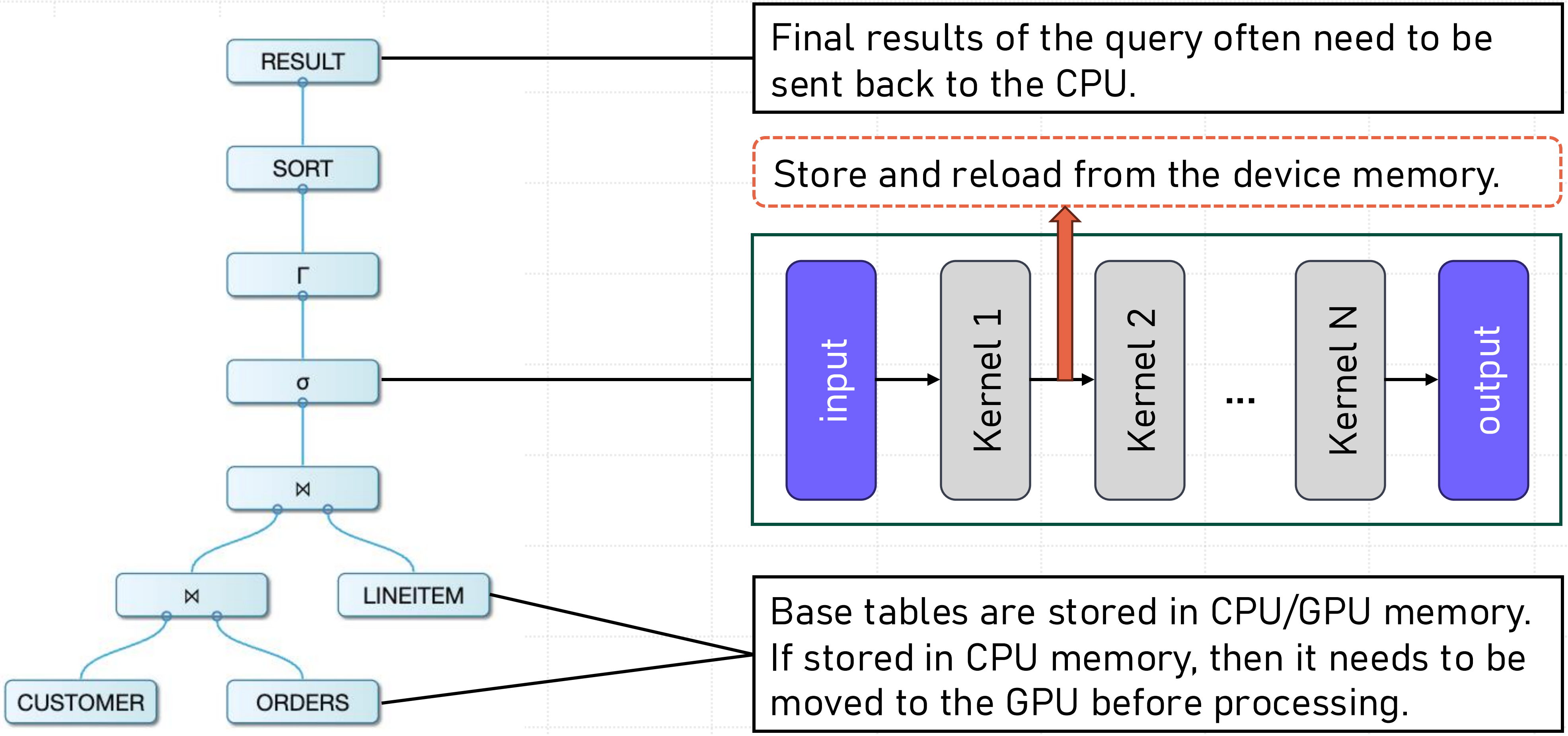
Current State

- **Research**
  - More than two decades of research
  - Crystal (fully in-GPU)
  - Operator studies
  - Maximus/Eiger (SG), Microsoft TQP, SiriusDB

- **Industry**
  - cuDF (NVIDIA), hipDF (AMD)
  - BlazingSQL, Kinetica, HeavyDB, Voltron Data (Startups)
  - Velox + Wave/cuDF (Meta)
  - IBM, Databricks, Microsoft, …



Introduction to GPU-based DB



Is the query exec bound by ***GPU*** or ***IC***?

- **MaxBench** – Single-GPU DB Benchmark
- Authors: Marko Kabić, Bowen Wu, Jonas Dann, Gustavo Alonso

Table 1: Hardware Configurations for the empirical evaluation.

Configuration	C_1 = PCIe3+A100	C_2 = PCIe5+H100	C_3 = GH200 (NVLink4+H200)	C_4 = PCIe4+RTX3090
CPU	Intel Xeon Platinum 8171M	AMD EPYC 9124	NVIDIA Grace	AMD EPYC 7313
CPU cores	2x6	2x16	72 ARM Neoverse V2 cores	2x16
GPU	NVIDIA A100 40GB	NVIDIA H100 80GB	NVIDIA H200 96GB	NVIDIA RTX3090 24GB
GPU Mem. Bandwidth	1.55 TB/s	4 TB/s	4 TB/s	0.936 TB/s
GPU clock (base–boost)	765–1410 MHz	1080–1785 Mhz	1980–1980 Mhz	1395-2100 MHz
Power cap	400W	400W	624.15W / 900W	350W
Interconnect (IC)	PCIe 3.0	PCIe 5.0	NVLink 4.0	PCIe 4.0
IC Bandwidth (1-way)	16 GB/s	64 GB/s	450 GB/s	32 GB/s

MaxBench – Single-GPU DB Benchmark

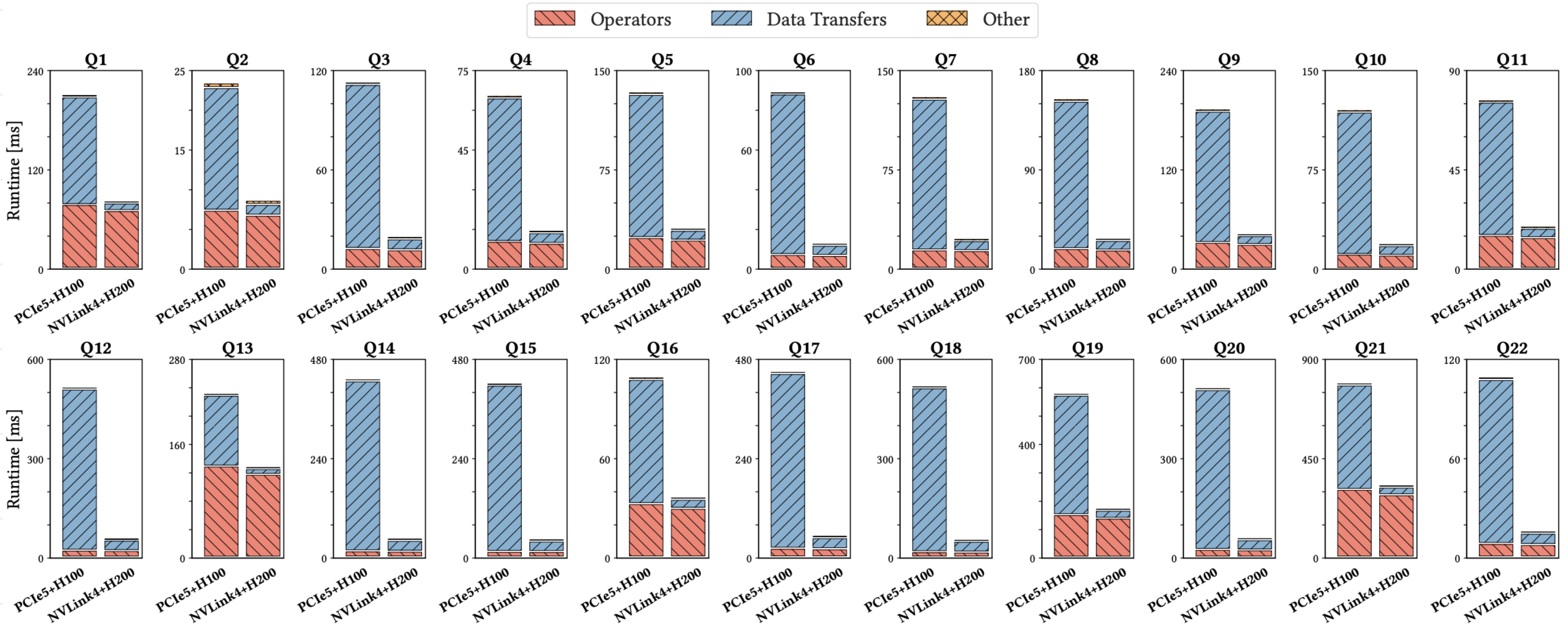
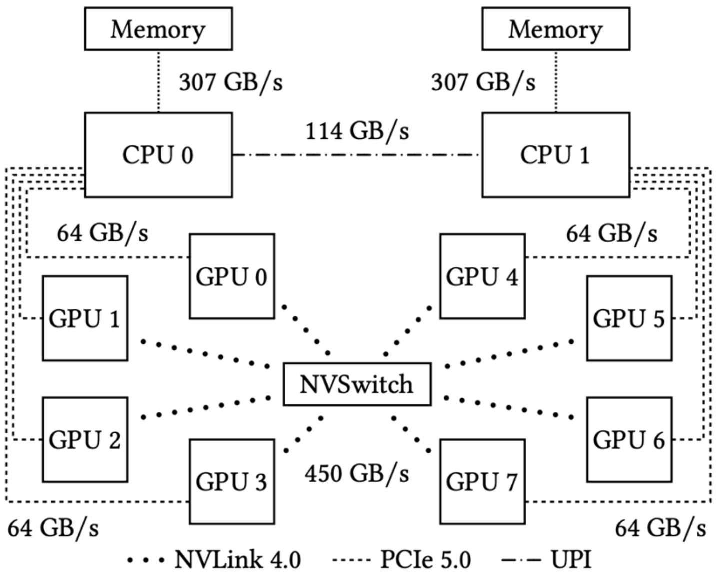
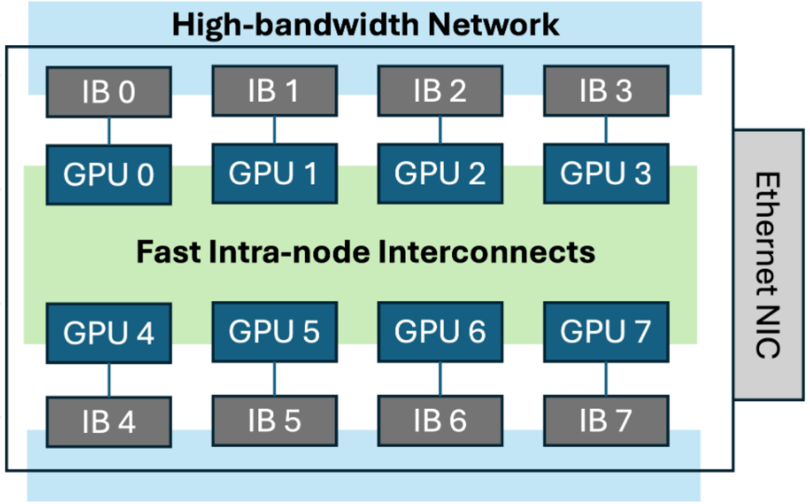


Figure 3: The full TPC-H benchmark (SF=10) run on hardware configurations C_2 = PCIe5 + H100 and C_3 = NVLink4+H200.

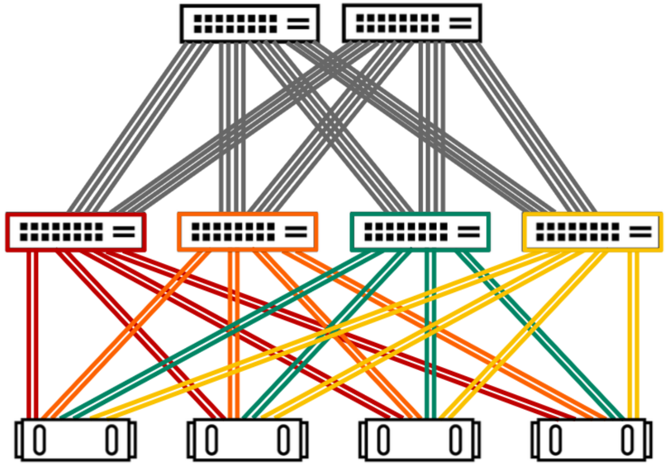
Distributed GPU cluster



8x H100 SXM5 80 GB[1]

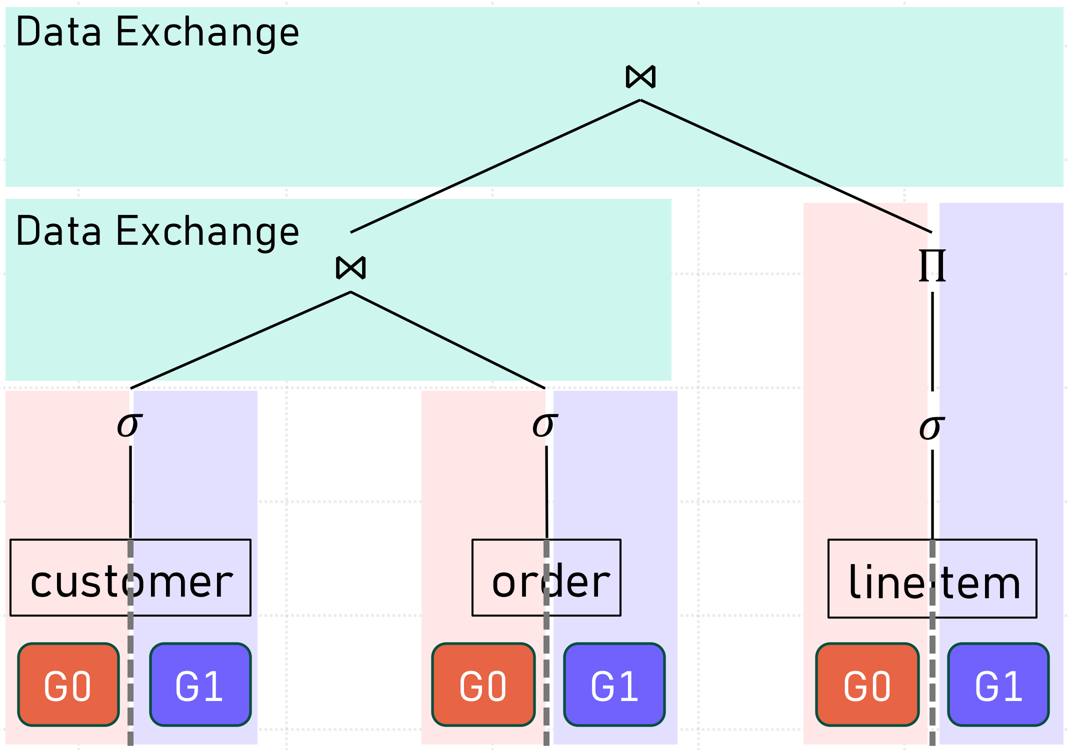


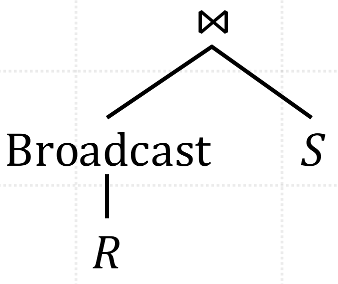
GPUs+NICs[2]

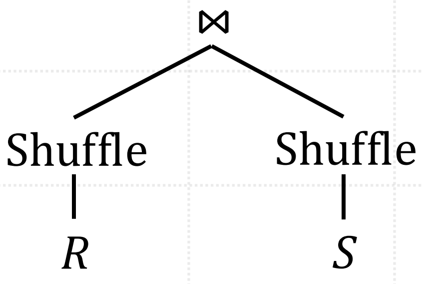


Switches[3]

Distributed DB







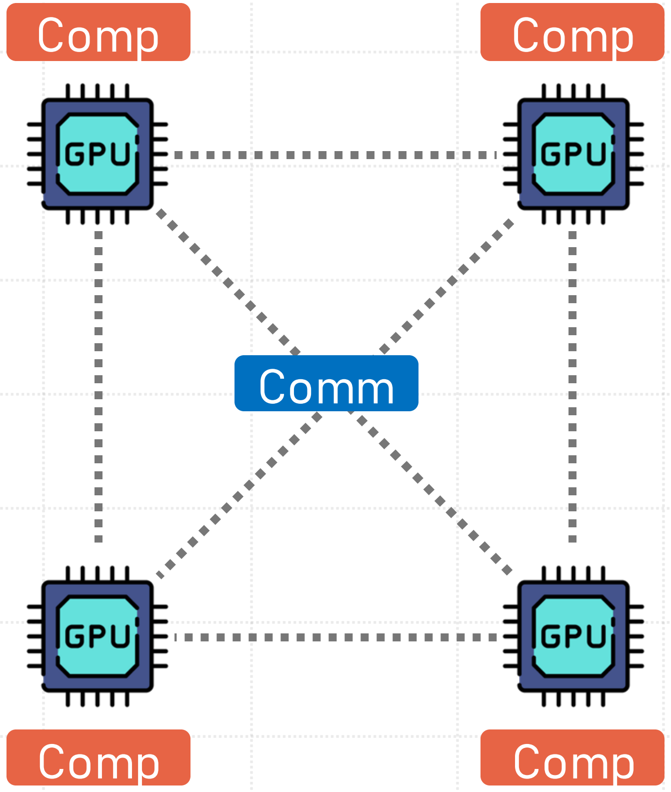
Broadcast

- Works well when the data is small.
- No extra overhead.
- Resistant to skew (later)

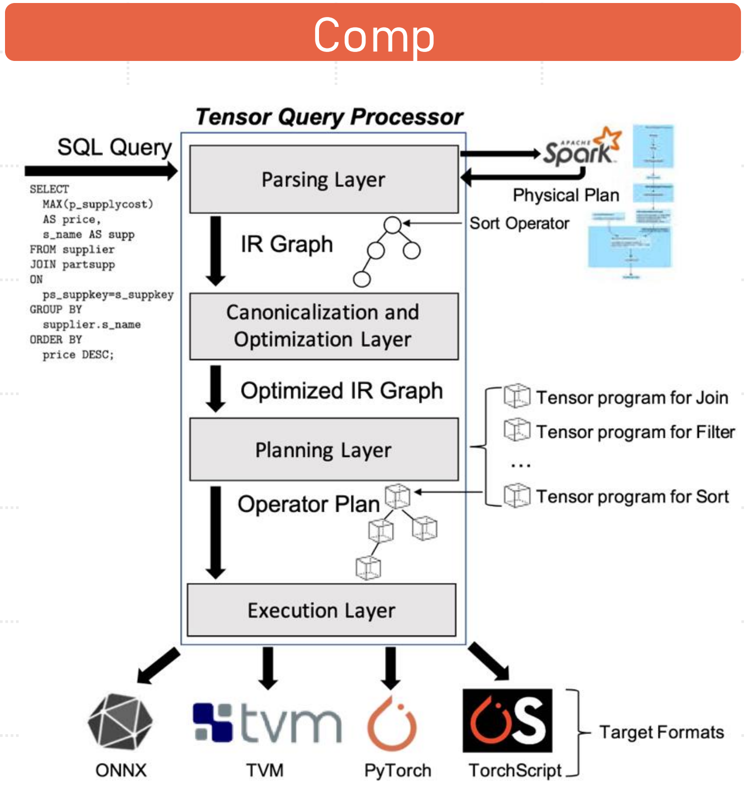
Shuffle

- Works well when the data is large.
- Extra overhead of partitioning data.
- Not very resistant to skew (later)

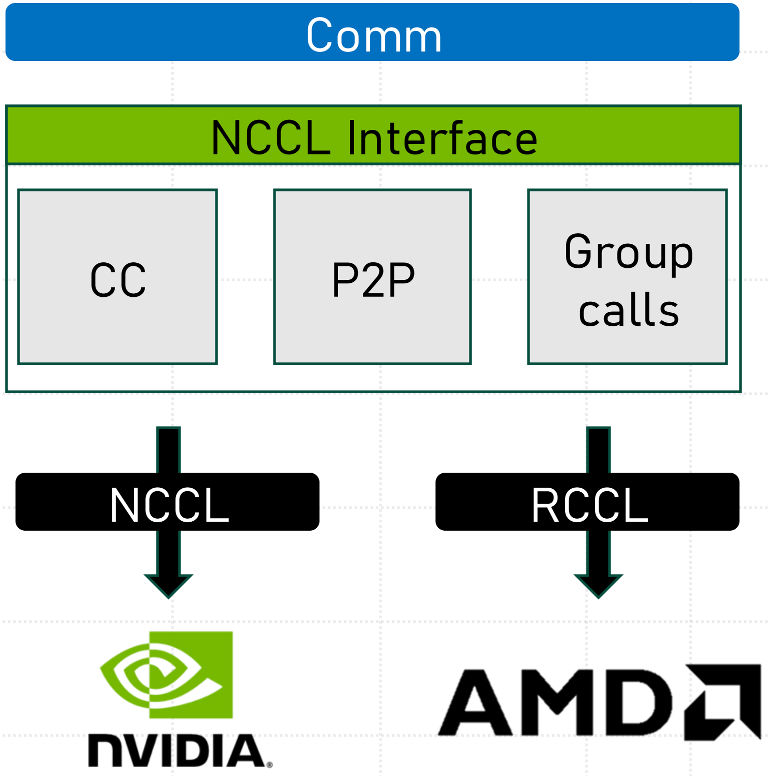
Our System



Comp
Comp
GPU
GPU
Comm
GPU
GPU
Comp
Comp



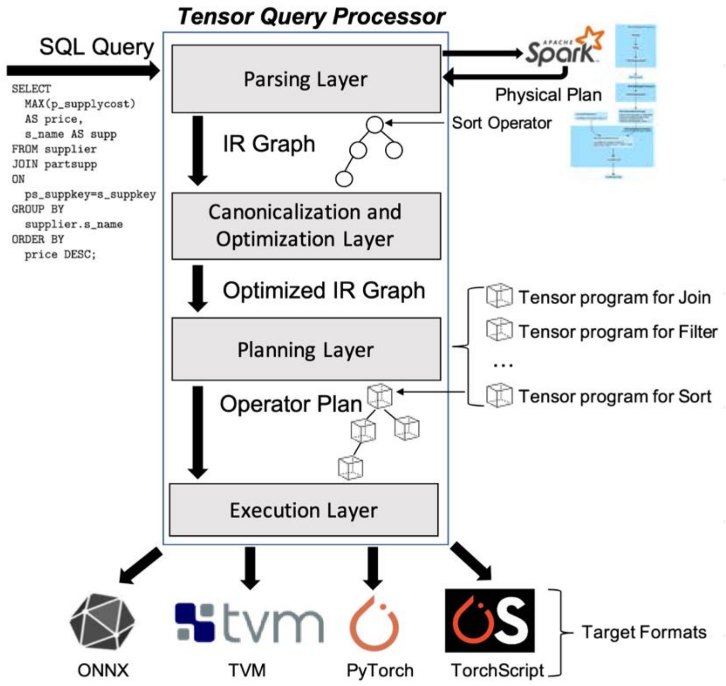
Comp
Tensor Query Processor
SQL Query
SELECT
MAX(p_supplycost)
AS price,
s_name AS supp
FROM supplier
JOIN partsupp
ON
ps_suppkey=s_suppkey
GROUP BY
supplier.s_name
ORDER BY
price DESC;
Parsing Layer
Spark
Physical Plan
Sort Operator
IR Graph
Canonicalization and
Optimization Layer
Optimized IR Graph
Tensor program for Join
Tensor program for Filter
...
Tensor program for Sort
Planning Layer
Operator Plan
Execution Layer
ONNX
TVM
PyTorch
TorchScript
Target Formats



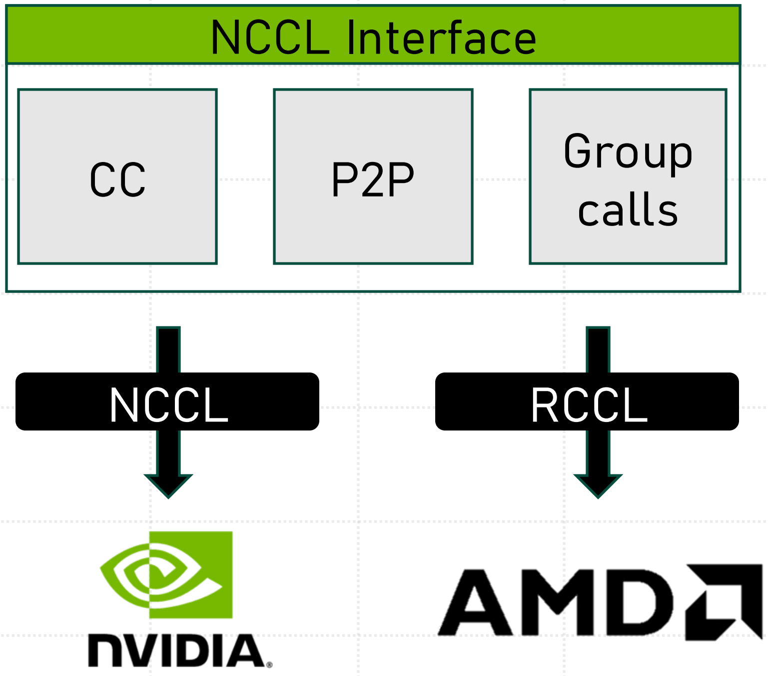
Comm
NCCL Interface
CC
P2P
Group
calls
NCCL
RCCL
NVIDIA
AMD

Our System

Comp



Comm



Using existing ML libraries gives

- Good out-of-box performance
- Portability
- Easy-to-develop

But it may not fully suit the DB workload, which often exhibits lots of ***irregularity***.

Research question:
How efficiently can we build a DB with GPU acceleration using ML libraries?

Experiment Setup

- We report the total time of all 22 TPC-H queries at SF=1000 or 3000.
- Data are already partitioned and loaded into the GPUs' memory.

Table 3: Cluster Configurations. Eth: Ethernet. IB: InfiniBand.

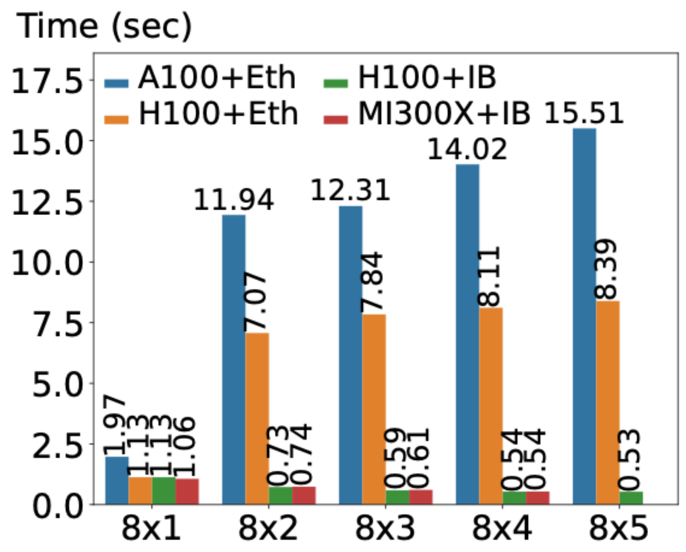
Cluster	GPU Type	HBM (GiB)	k	V	GPU Interconnect Intra-VM	GPU Interconnect Inter-VM	CPU type	CPU Cores	CPU Mem (GiB)	Price/hour (USD)
1	NVIDIA A100	80	8	7	NVLink 300 GB/sec	Eth: 1x50 Gbits/sec	AMD EPYC 7V12	96	1800	32.77*
2	NVIDIA H100	79.6		5	NVLink 450 GB/sec	Eth: 1x100 Gbits/sec IB: 8x400 Gbits/sec	Intel Xeon Platinum 8480C		1900	98.32
3	AMD MI300X	191.5		4	Infinity Fabric 448 GB/sec	Eth: 1x100 Gbits/sec IB: 8x400 Gbits/sec	Intel Xeon Platinum 8480C		1850	63.6

* This is the price for 8x200 Gbits/sec Infiniband. The Eth version, where we run our experiment, is not publicly listed.

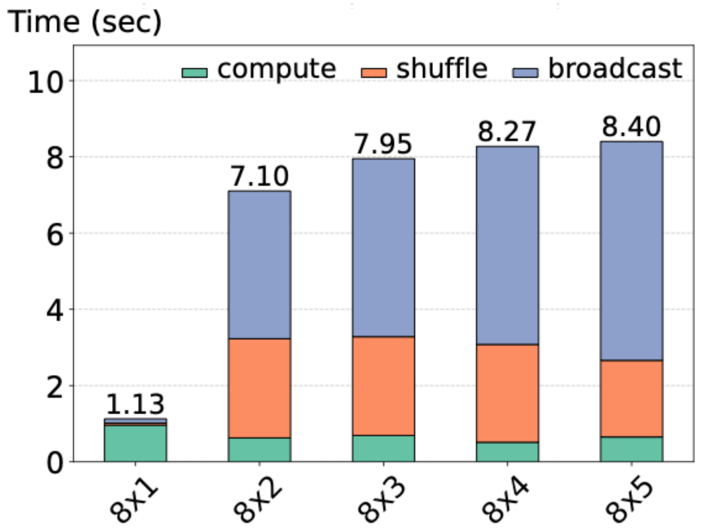
Key Results

How will the performance change if my cluster changes or my workload changes?

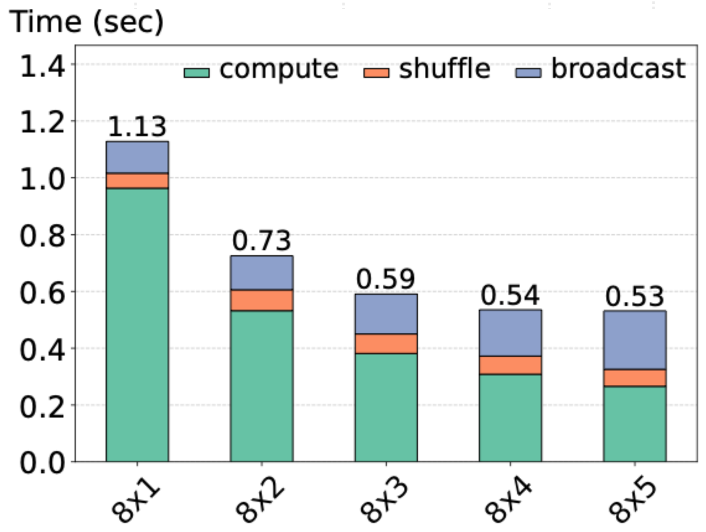
- Scale-out (more machines)?
- Scale-up (more GPUs/machine, faster NVLink and network)?
- Workload sensitivity (skew, data placement, etc.)



- GPU-based DBs can be 1-2 orders of magnitude faster than CPU-based ones.



- A fast network is necessary for good scalability.



- With a fast network, the query exec transitions from GPU-bound to network-bound as the cluster grows.

Effect of *data skew*

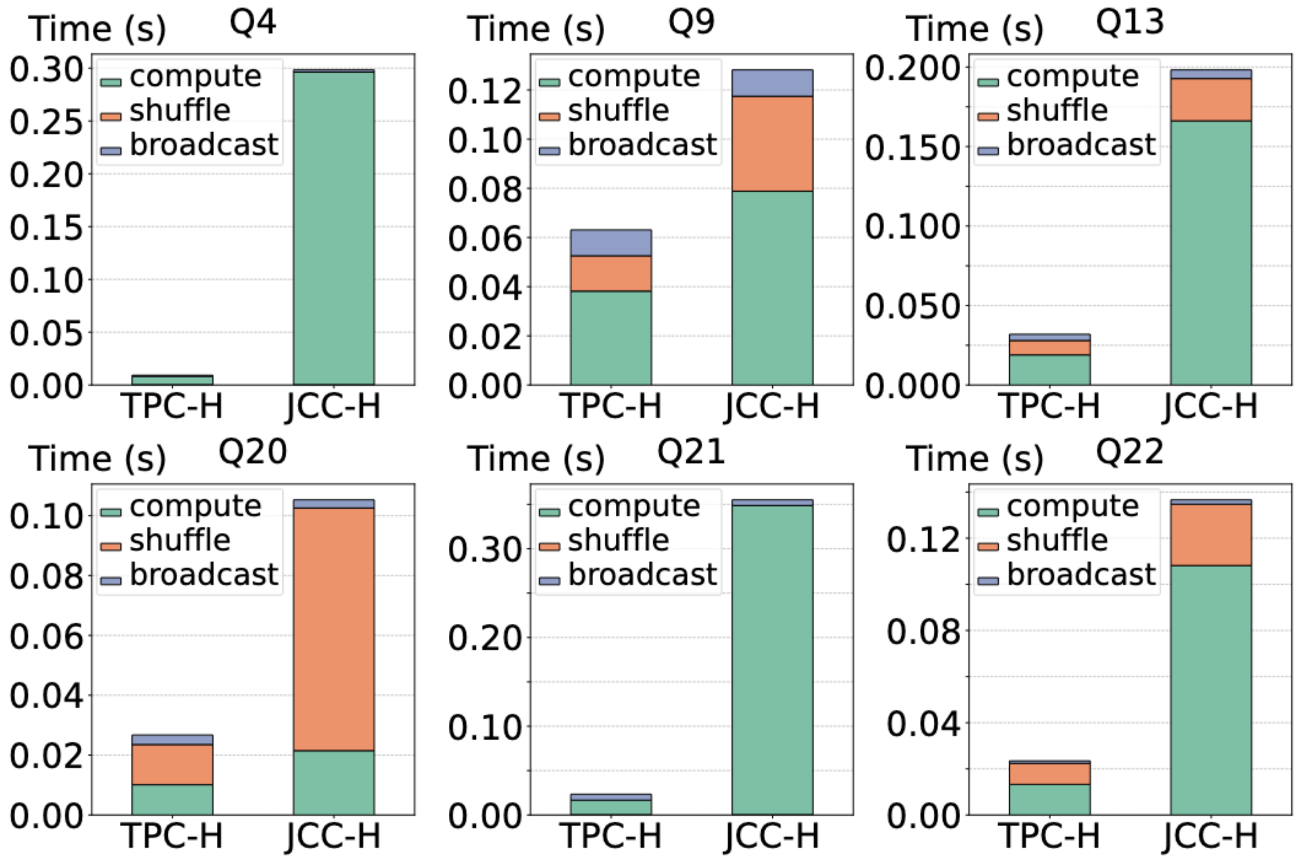


Figure 21: Time breakdown comparison. (V=5)

JCC-H 1TB
Same schema + same query + skewed data

Key Contributions

How will the performance change if my cluster changes or my workload changes?

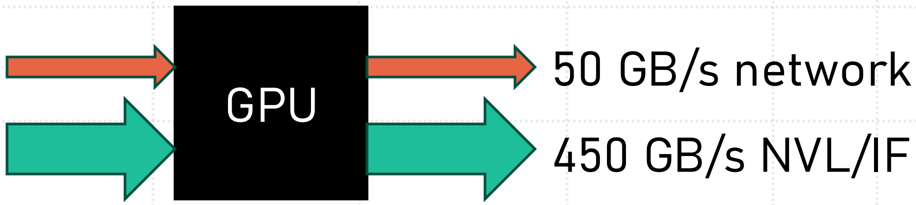
- Scale-out (more machines)?
- Scale-up (more GPUs/machine, faster NVLink and network)?
- Workload sensitivity (skew, data placement, etc.)

Model the computation and communication of query execution

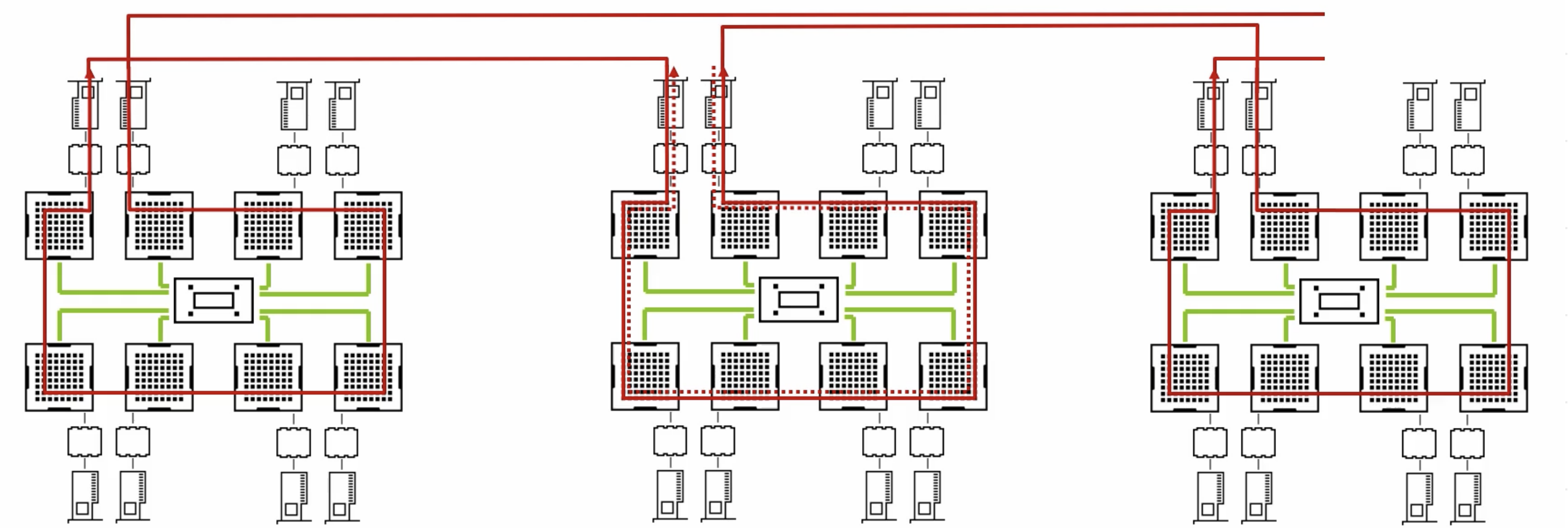
Challenges:

- The distributed GPU cluster has a very heterogeneous interconnect.
- The algorithm used by NCCL is obscure.
- Previous effort main focused on modeling ML workloads.

Modeling Broadcast



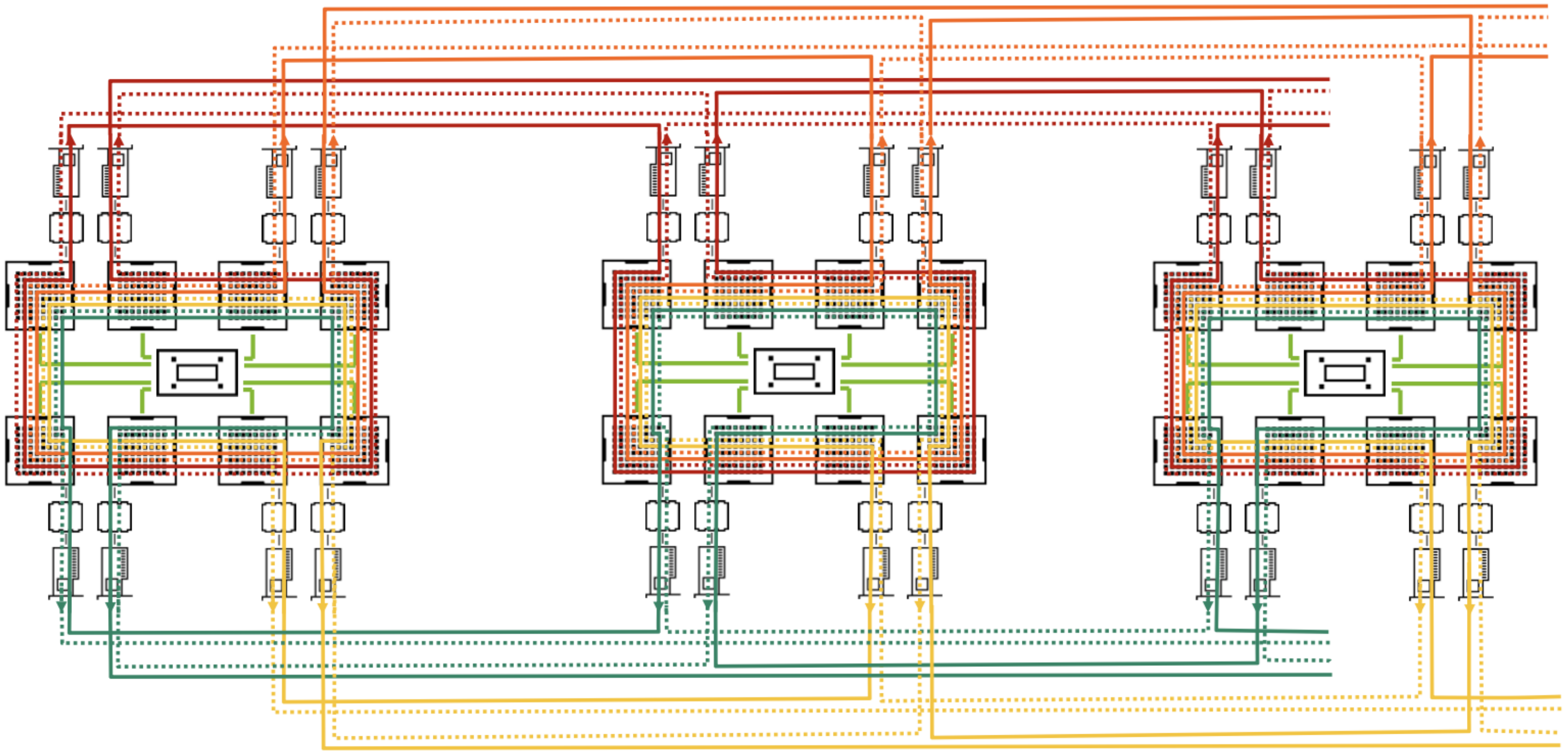
Formation of a single ring



https://www.nvidia.com/en-us/on-demand/session/gtc24-s62129/

Modeling Broadcast

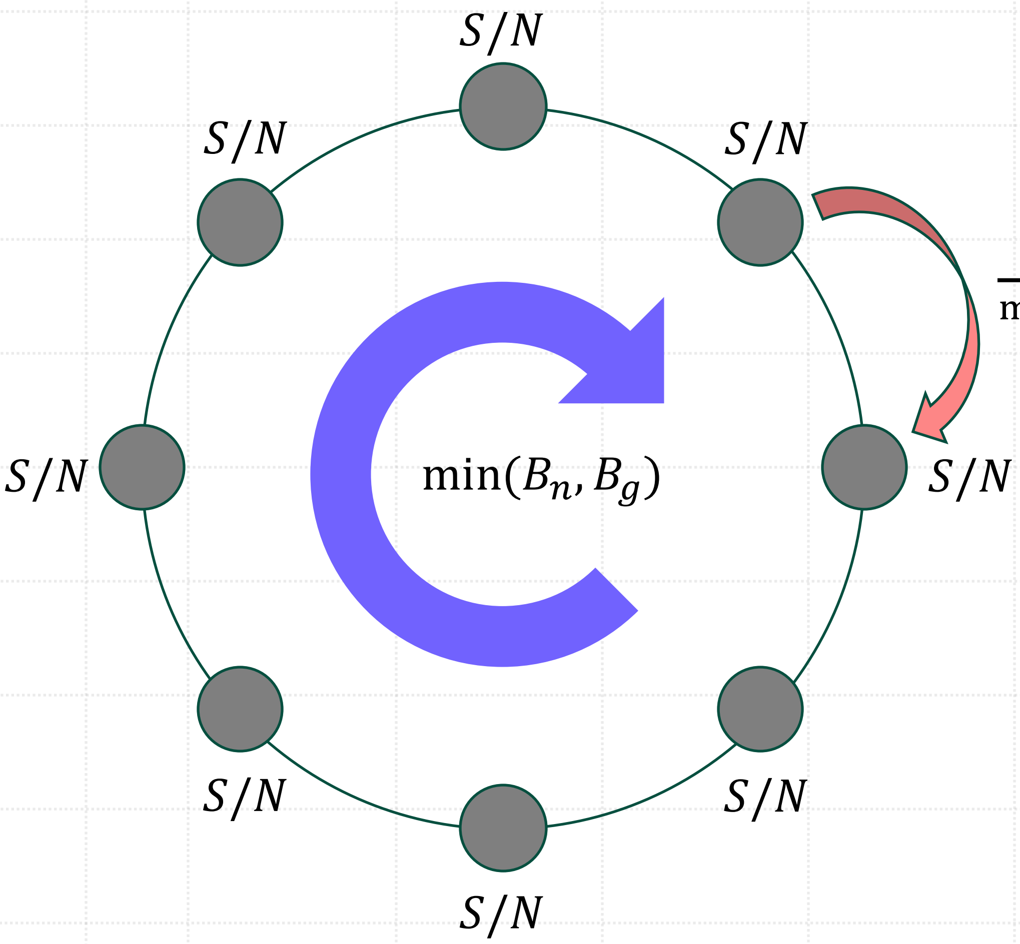
Formation of all rings



https://www.nvidia.com/en-us/on-demand/session/gtc24-s62129/

k	Number of GPUs per machine (or node, VM)
V	Number of machines (or nodes, VMs)
N	Total number of GPUs = $k \times V$
$\boldsymbol{B_g}$	Unidirectional inter-GPU bandwidth within each machine
$\boldsymbol{B_n}$	Unidirectional network bandwidth at each machine
S	Total dataset size processed by the GPU cluster.
$\boldsymbol{G_{ij}}$	The i-th GPU in the j-th machine.
$\boldsymbol{m_{ij \to pq}}$	The message sent from G_{ij} to G_{pq}.

Modeling Broadcast



$\frac{S/N}{\min(B_n,B_g)}$ time/step

In total, you need $(N-1)$ steps.

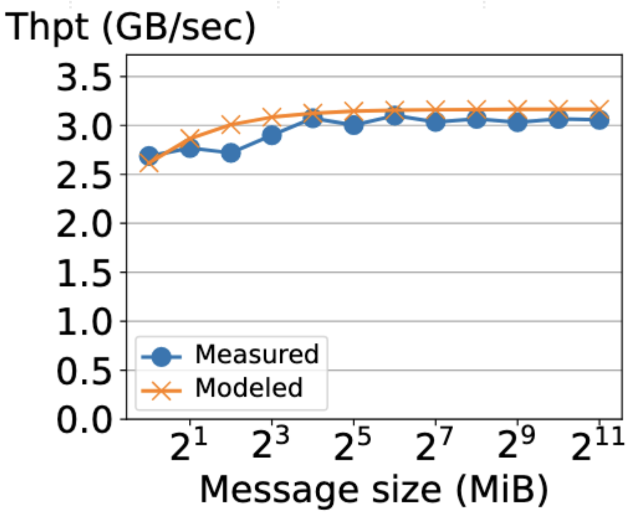
➔ Total time = $(N-1)\frac{S/N}{\min(B_n,B_g)}$

➔ Throughput = $\frac{N}{N-1}\min\left(B_n, B_g\right)$

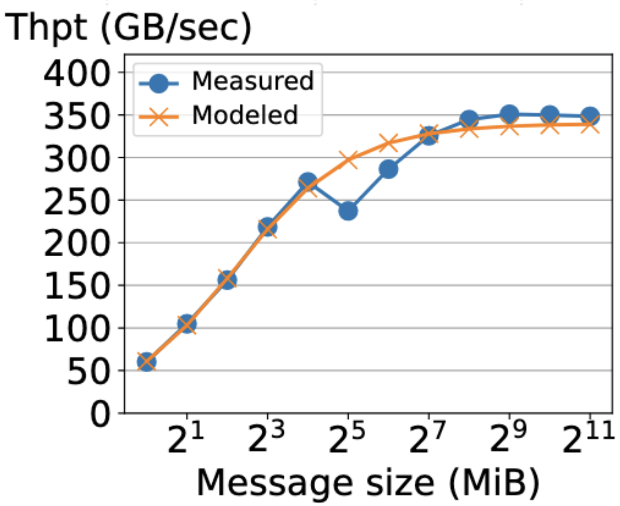
Takeaways:

- Throughput ***decreases*** with #GPUs.
- Faster network won't improve the performance.

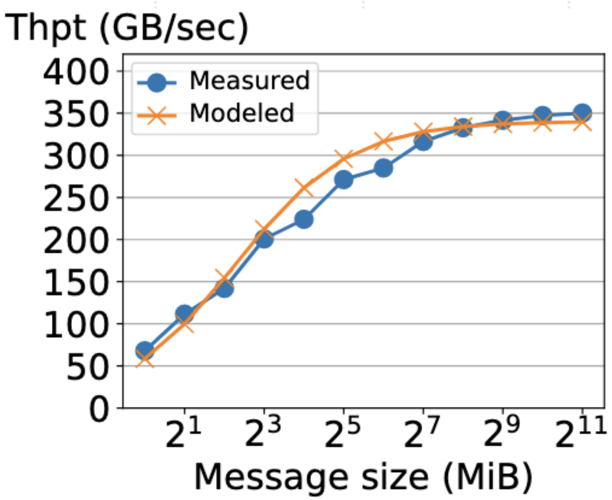
Modeling Broadcast



(a) Broadcast (H100+Eth).



(c) Broadcast (H100+IB).



(e) Broadcast (MI300X+IB).

k	Number of GPUs per machine (or node, VM)
V	Number of machines (or nodes, VMs)
N	Total number of GPUs = $k \times V$
$\boldsymbol{B_g}$	Unidirectional inter-GPU bandwidth within each machine
$\boldsymbol{B_n}$	Unidirectional network bandwidth at each machine
S	Total dataset size processed by the GPU cluster.
$\boldsymbol{G_{ij}}$	The i-th GPU in the j-th machine.
$\boldsymbol{m_{ij \to pq}}$	The message sent from G_{ij} to G_{pq}.

Modeling Broadcast with Skew

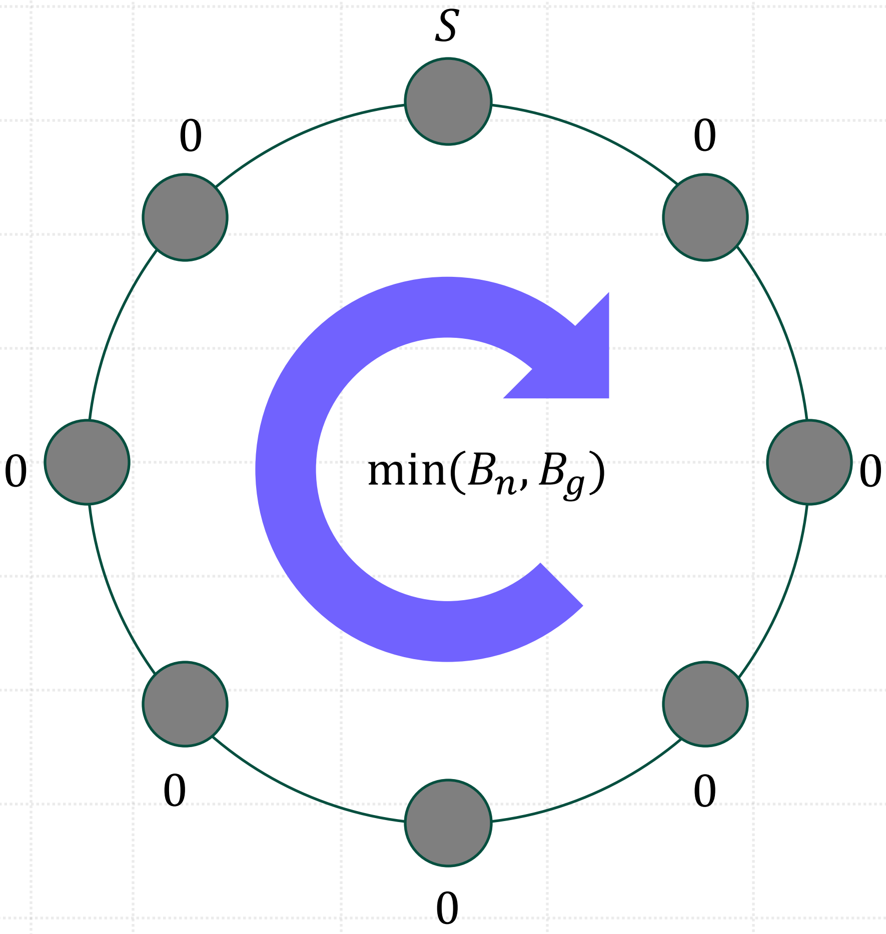


TABLE I (from https://arxiv.org/abs/2507.04786)
COMPARISON OF NCCL COMMUNICATION PROTOCOLS

	Simple	**LL**	**LL128**
Design Goal	High bandwidth	Low latency	Low latency and high bandwidth
Synchronization Mechanism	Memory fences (high overhead)	Flag-based synchronization	Flag-based synchronization
Payload	Data chunks	4B data + 4B flag	120B data + 8B flag
Bandwidth Utilization	Near peak	$25 \sim 50\%$ of peak [11]	$\sim 95\%$ of peak [11]
Latency Per-hop	$\sim 6\mu s$	$\sim 1\mu s$	$\sim 2\mu s$

Due to pipelining and ring formation, the skewed data distribution has a minimal impact on performance, except for a slight increase in latency.

Model Skew

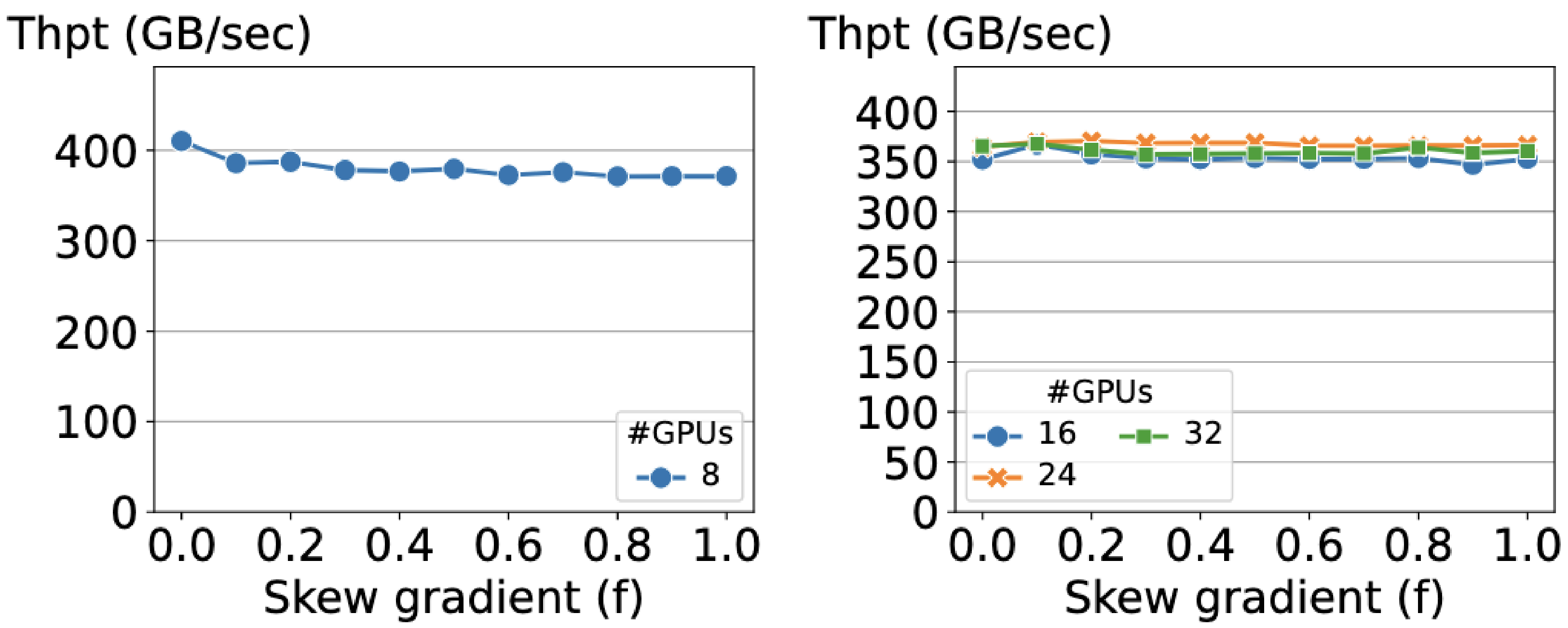
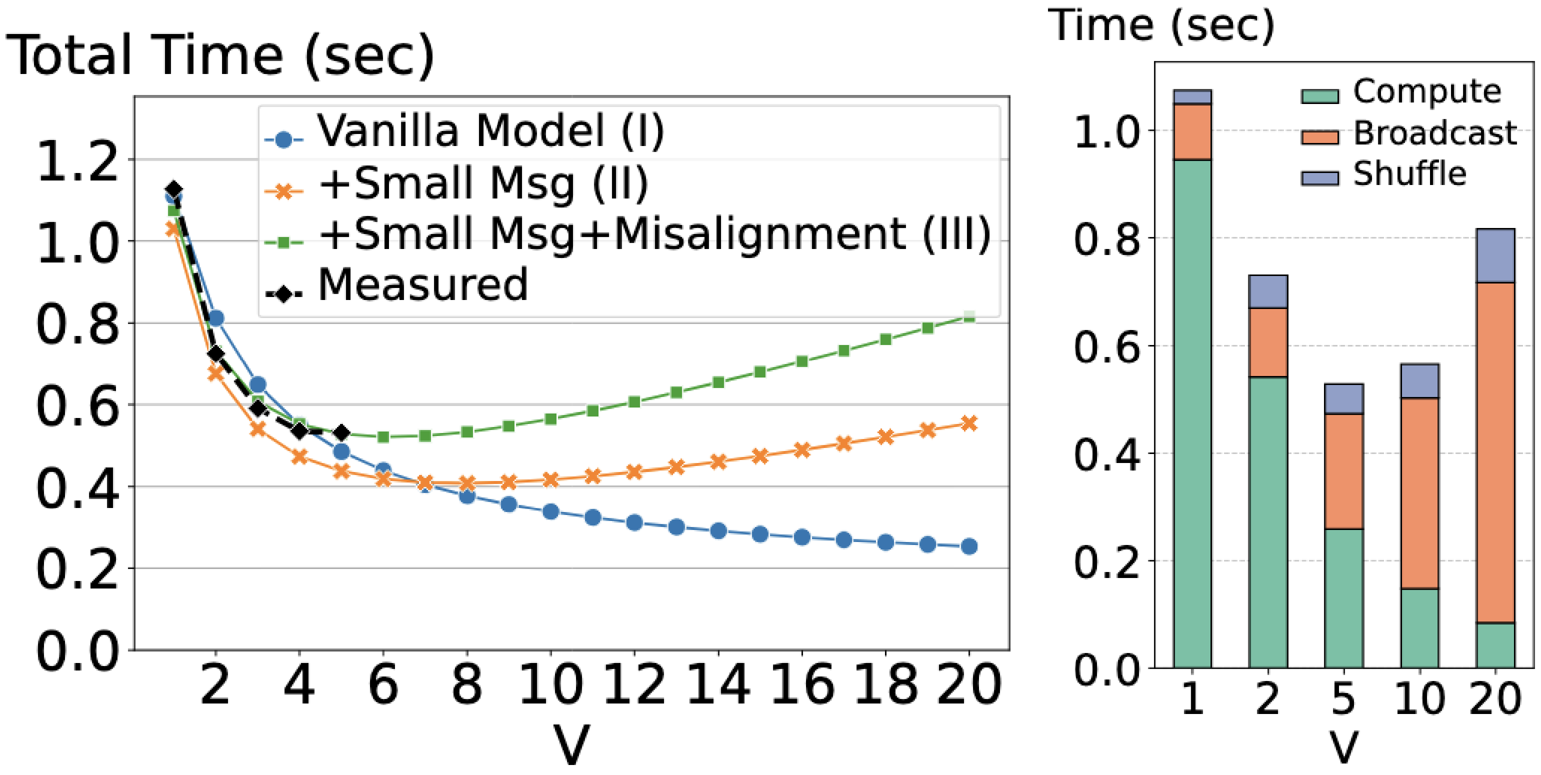


Figure 8: Broadcast + data skew.

Model TPC-H Queries



(a) Models for TPC-H projections. **(b) Project breakdown.**

Key Takeaways

- Using ML-based software to implement DBs gives good performance when the problem size is large enough.
- Small message sizes, skew, and buffer misalignment are the biggest sources of inefficiencies.
- Together with MaxBench, we see that with fast CPU-GPU, GPU-GPU, and GPU-network interconnects, the query performance often becomes GPU-bound.

  ➔ This calls back a previous work we published: "*Efficiently Processing Joins and Grouped Aggregations on GPUs*", SIGMOD 2025.

Next Steps