Multiverse

Accelerating Design Space Exploration for LLM Training Systems with Multi-experiment Parallel Simulation

Fei Gui, Kaihui Gao, Li Chen, Dan Li, Vincent Liu, Ran Zhang,
Hongbing Yang, Dian Xiong

The current surge of LLM infrastructure investments

Analysts forecast that the **LLM training infrastructure** market will reach $222.4 billion by 2030, with a compound annual growth rate (CAGR) of 25.5%.

19B

Total investment for 2024

Microsoft

Microsoft is part of the Global AI Infra. Investment Partnership aiming to raise $30 billion initially, with a long-term goal of $100 billion for AI data centers and energy projects.

16.4B

In AI infrastructure for Q2 2024

aws Amazon

AWS plans to invest ~£8 billion (~$10.45 billion) over the next five years to build and operate data centers in the UK to enhance its cloud and generative AI(LLM) capabilities.

>38B

for the fiscal year 2024

Meta

Meta has set a budget for capital expenditures between $38 billion and $40 billion for the fiscal year 2024, with a notable focus on AI infra.

Design space exploration can be used to justify the investment

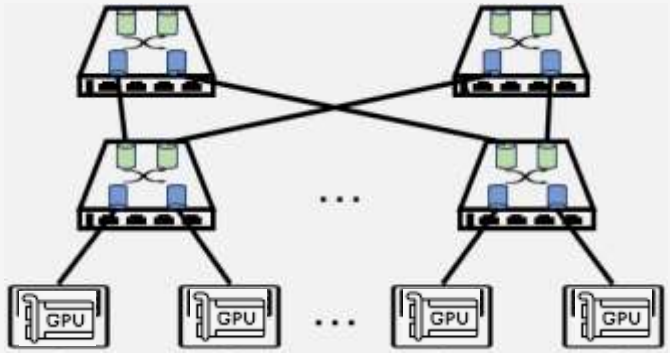
To justify the investment, we need to explore various design options.

- **[Planning]** What is the most suitable interconnection topo. given the current GPU scale?
  - ✓ Topology search: layers, port numbers, connection relationships.
- **[Operating]** What hyper-parameters to use for training?
  - ✓ Hyper-parameters search: the parallelization strategy, parameters of collective communication primitives, congestion control algorithms and parameters, etc.

The procedure can be defined as **Design Space Exploration**.

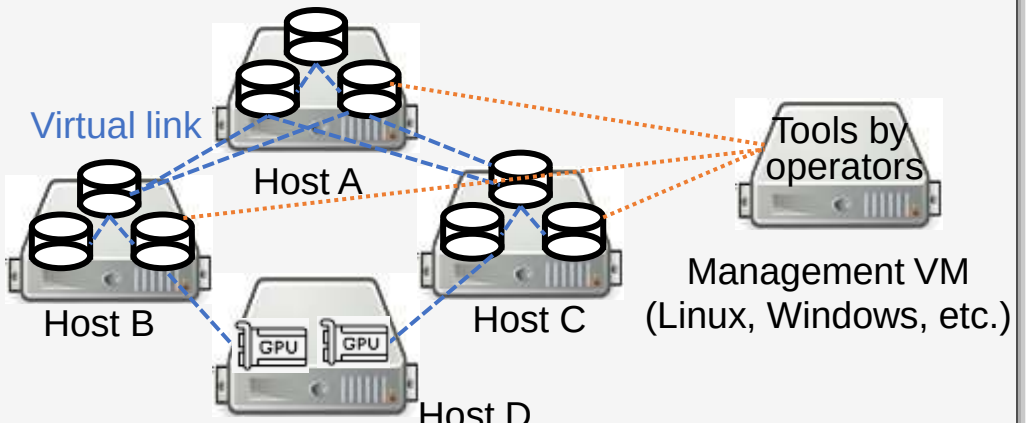
Simulator is the best choice in the design space exploration

- Hardware testbed
  - High-cost
  - Not flexible

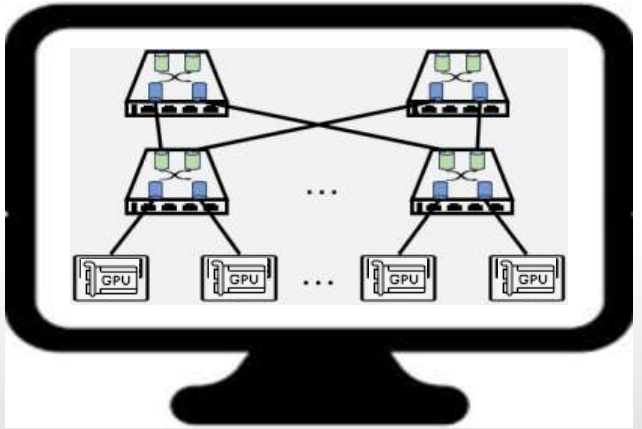


GPU cluster

- Emulator: Mininet, CrystalNet, etc.
  - High-cost



As a low-cost, flexible tool, simulator is the best choice in design space exploration.

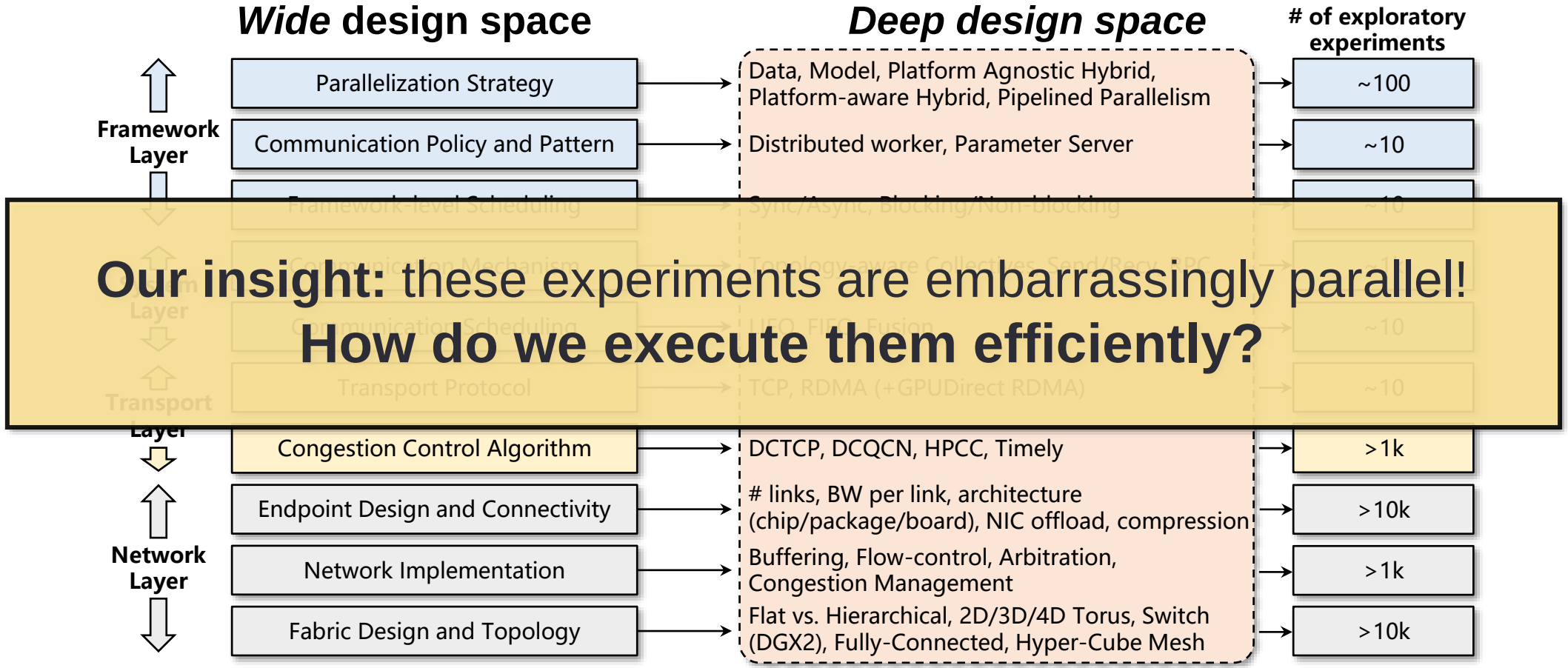


The design space exploration requires a huge number of simulation experiments

GPT-4 is trained by a cluster with ~25k GPUs

Gemini 2.0 is trained on a cluster with ~100k TPUs

XAI builds a cluster with ~200k GPUs to train Grok3



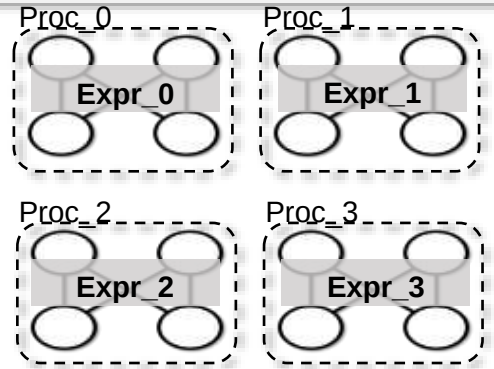
Our insight: these experiments are embarrassingly parallel!

How do we execute them efficiently?

How to accelerate the execution of embarrassingly parallel experiments

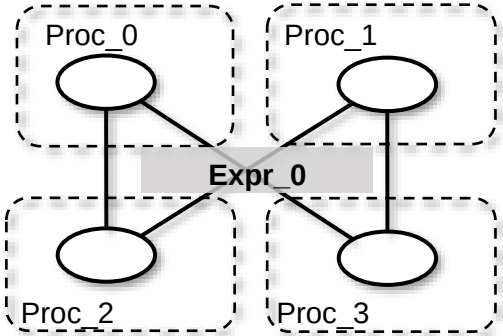
Existing simulators are not optimized for the execution of embarrassingly parallel experiments.

Single-experiment is accelerated by a **single process** with multi-thread
e.g. NS-3, OMNet++, UNISON, DONS



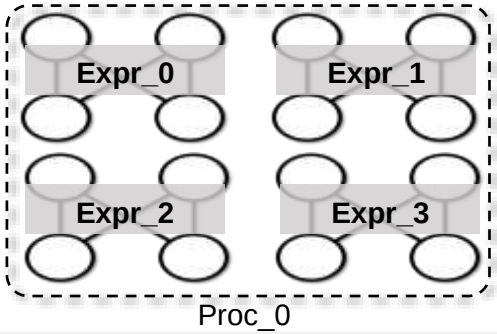
High context switching cost

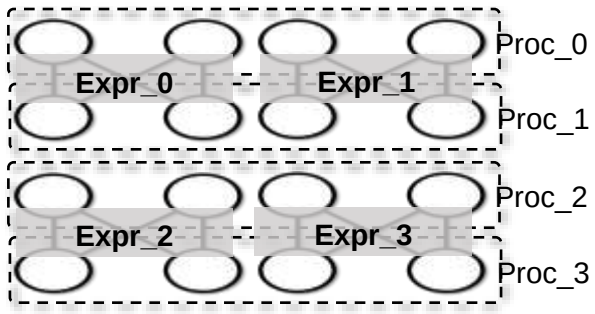
Single-experiment is accelerated by **Multi-process**
e.g. NS-3+MPI



High sync. overhead

Low context switching cost and sync. overhead



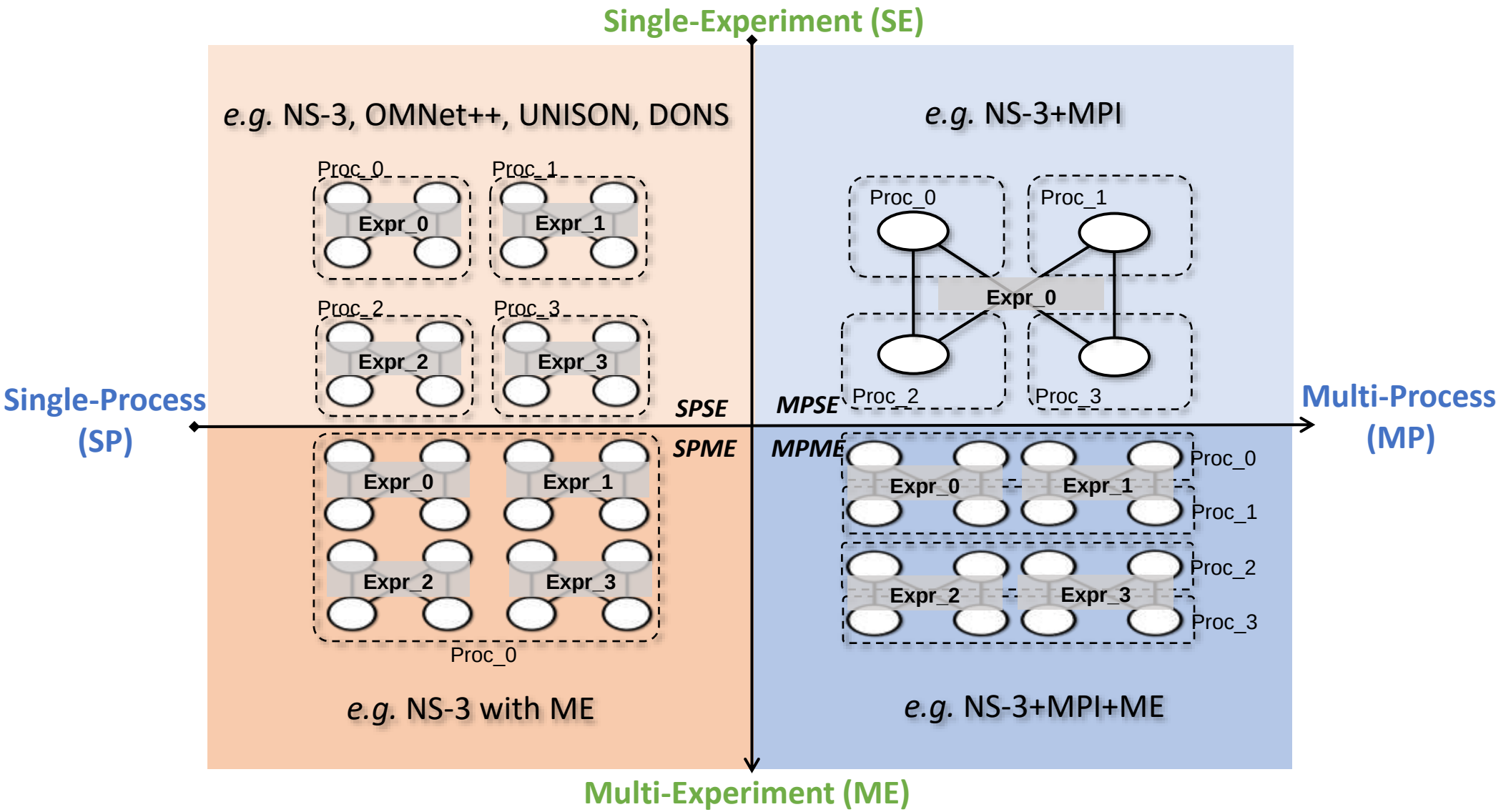


high cache miss rate

Multi-experiment are accelerated by a **single process** with Multi-thread
e.g. NS-3 with Multi-experiment

Multi-experiment are accelerated by **multi-process**
e.g. NS-3+MPI+Multi-experiment

Quadrants of parallelization strategies



Experiment setting for comparing four parallel strategies

Comparison

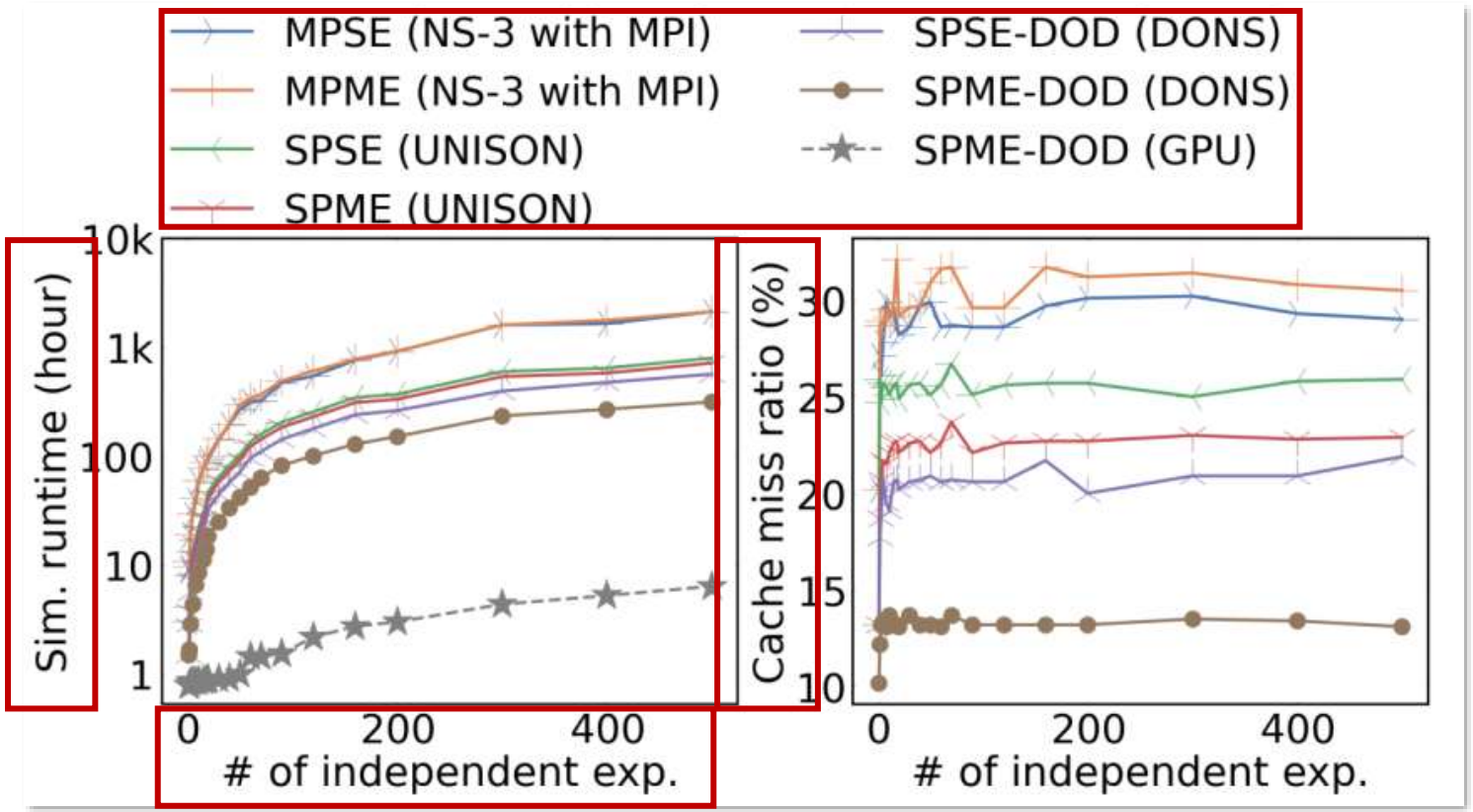
These four parallel strategies based on existing simulation technologies

Scenario

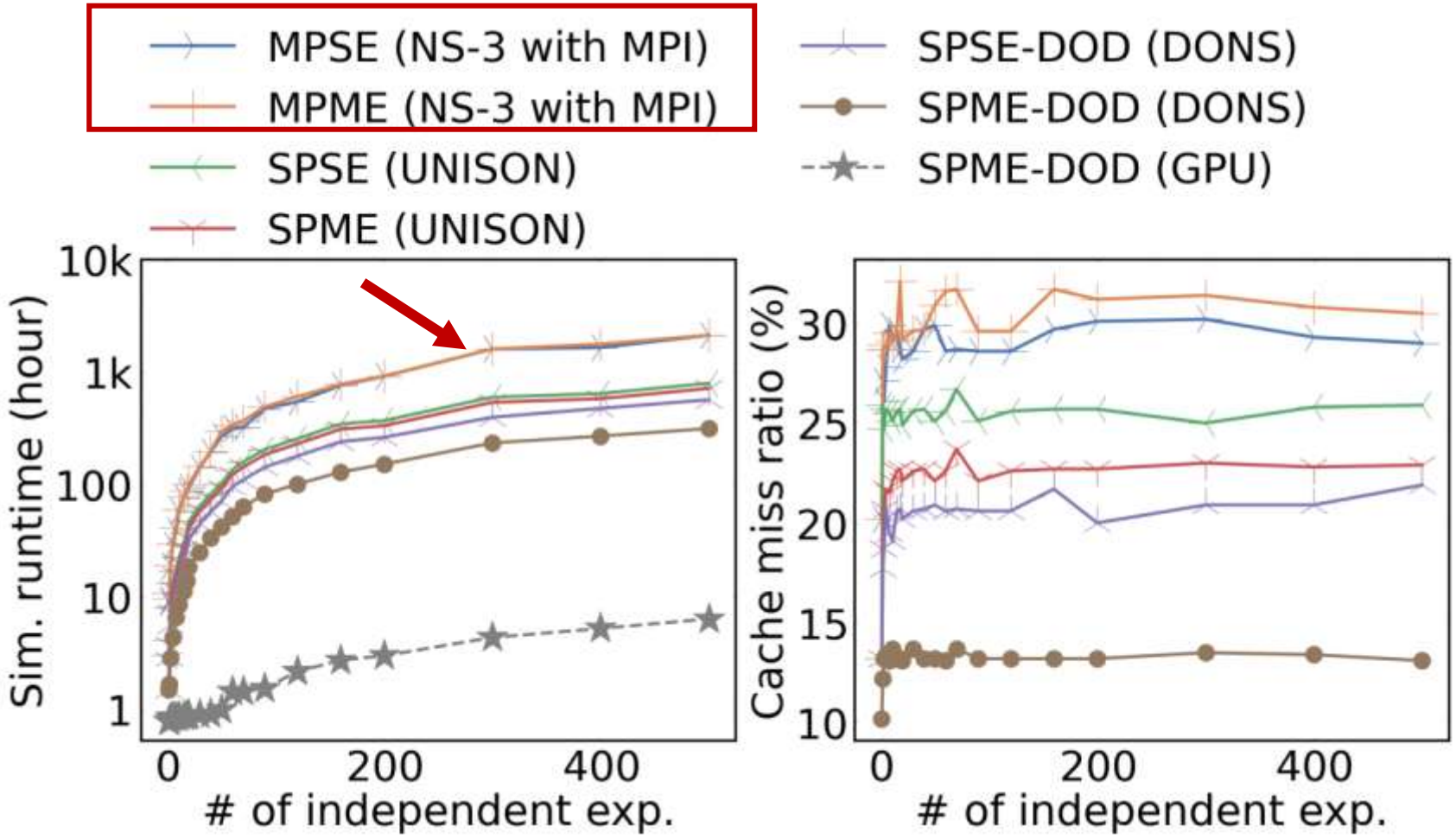
Searching for the optimal collective comm. algorithm for a cluster with 128 GPUs, training 13B model

Metric

1. the time to complete 1 to 500 independent sim. experiments
2. cache miss rate

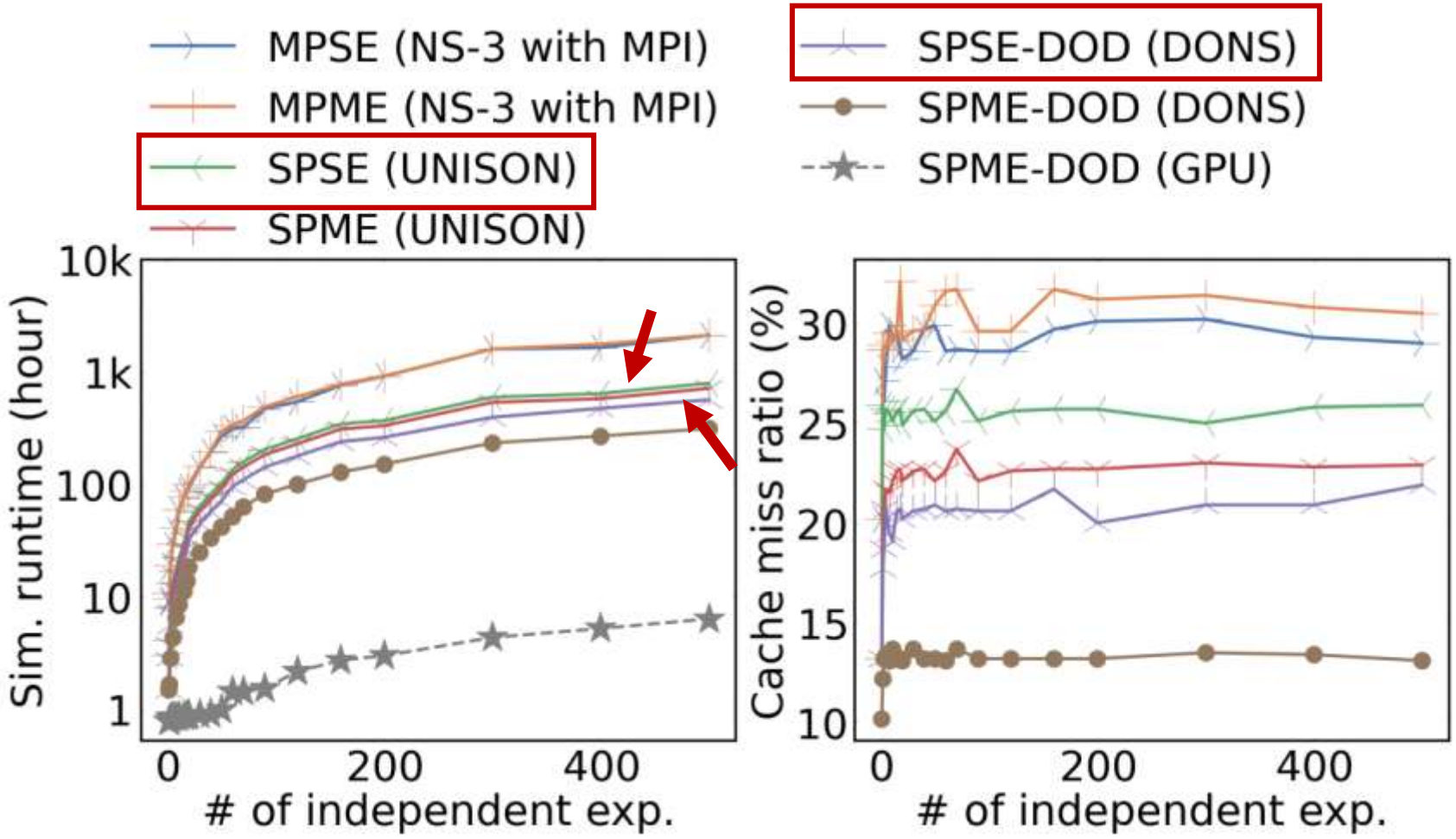


Experiment results



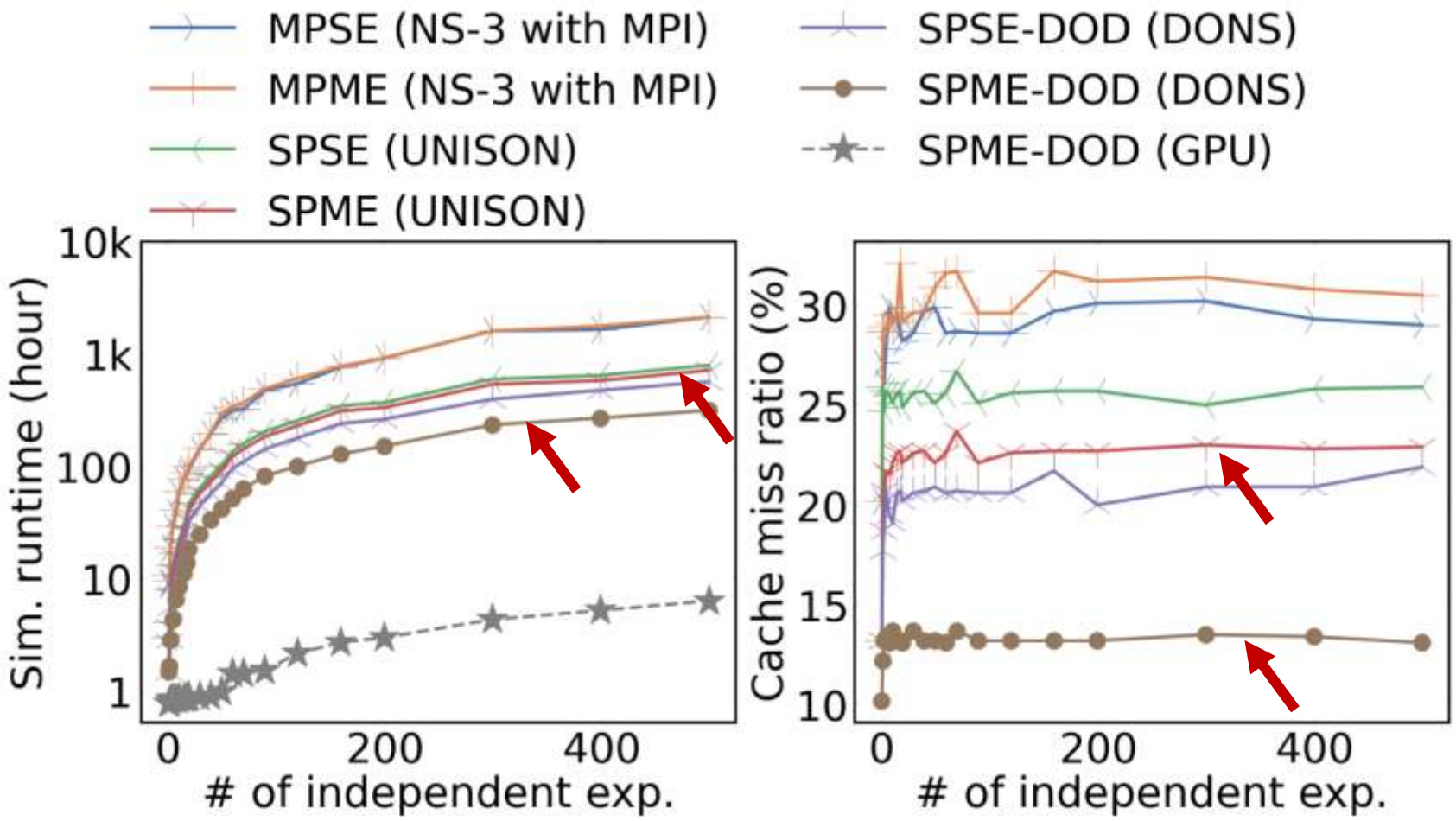
- **MP*E performs worst:** MP results in high synchronization overhead.

Experiment results



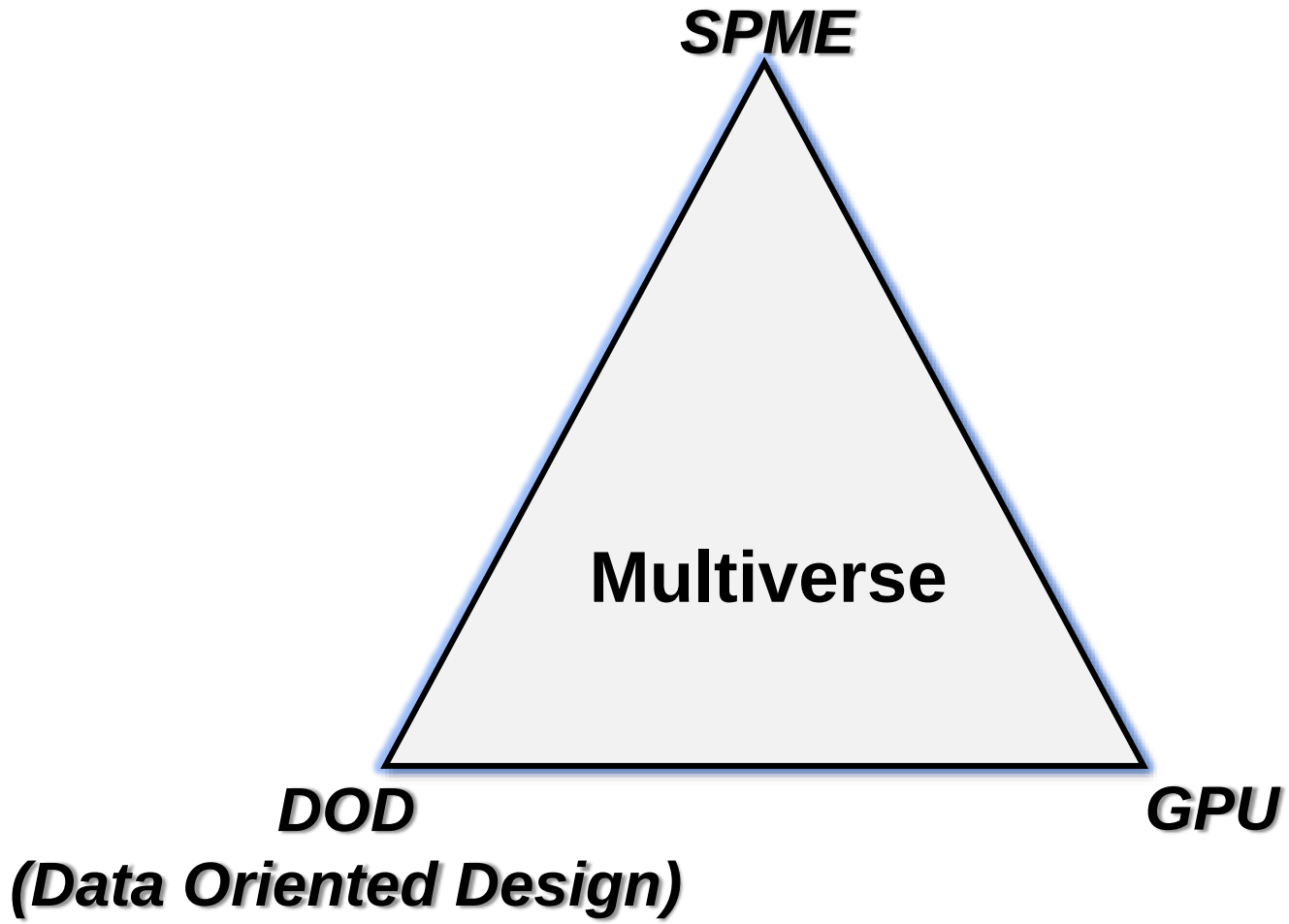
- **MP*E performs worst:** MP results in high synchronization overhead.
- **SPSE performs poorly:** high cache miss rate caused by Context Switching among experiments

Experiment results

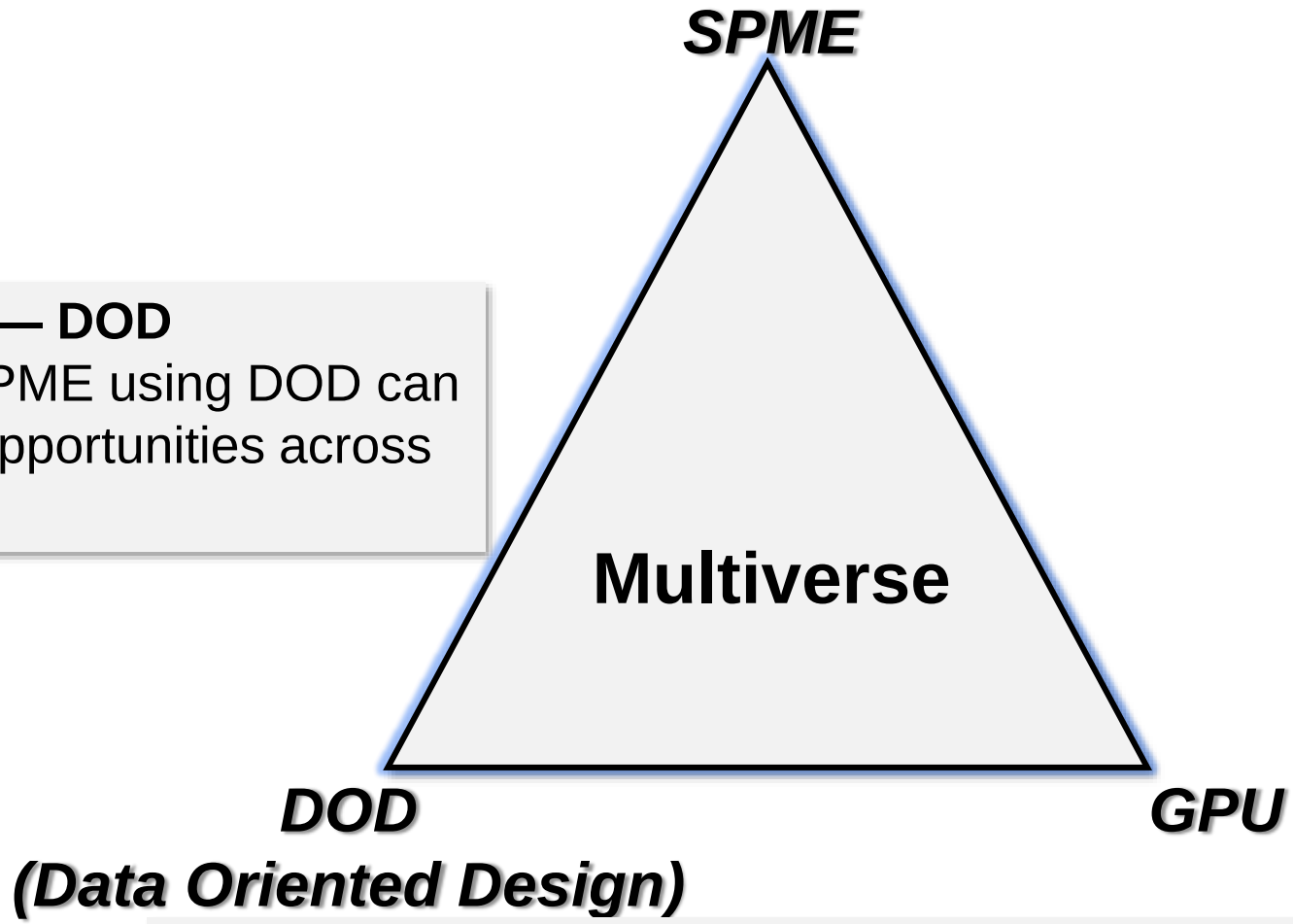


- **MP*E performs worst:** MP results in high synchronization overhead.
- **SPSE performs poorly:** high cache miss rate caused by Context Switching among experiments
- **SPME performs best, but still requires >370 h to finish all 500 experiments!**

How to unleash the potential of SPME?



How to unleash the potential of SPME?



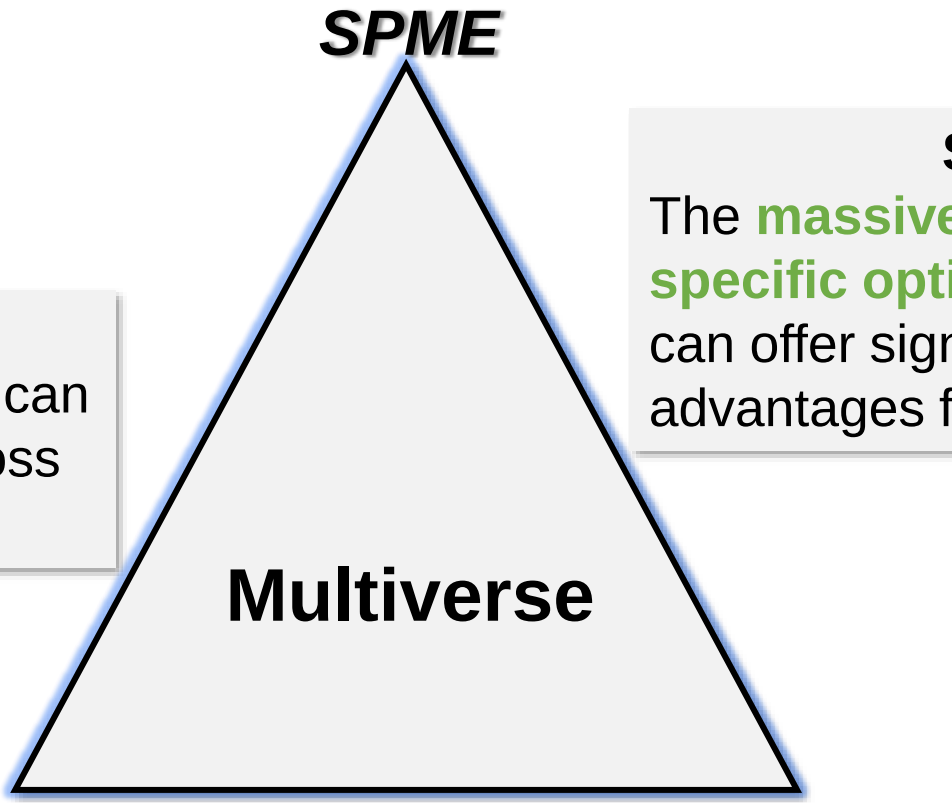
SPME — DOD

Implementing SPME using DOD can exploit parallel opportunities across experiments.

DOD — GPU

SPME + DOD aligns well with the **Single Instruction, Multiple Data (SIMD)** tasks that GPUs excel at.

How to unleash the potential of SPME?



SPME — DOD

Implementing SPME using DOD can exploit parallel opportunities across experiments.

SPME — GPU

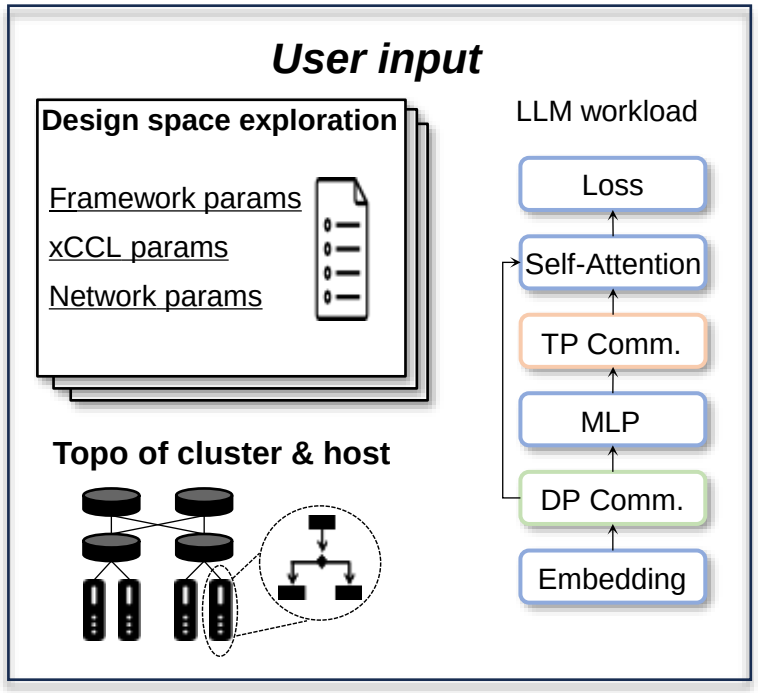
The **massive cores** in a GPU and its **specific optimizations for SIMD** tasks can offer significant performance advantages for SPME.

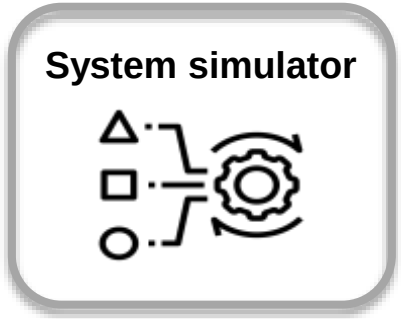
DOD — GPU

SPME + DOD aligns well with the **Single Instruction, Multiple Data (SIMD)** tasks that GPUs excel at.

Multiverse architecture

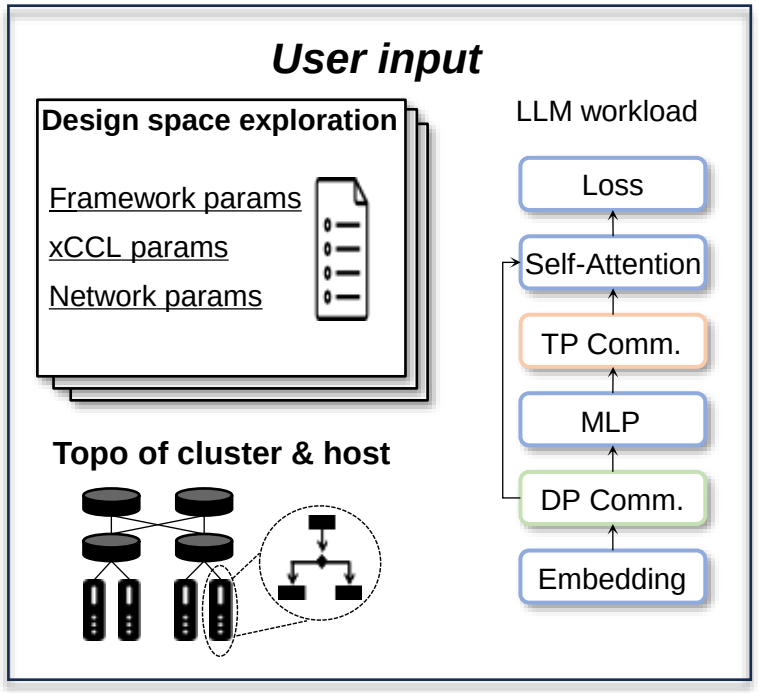
System Simulator: Controls and schedules the process of AI training.





User input: Chakra

Multiverse architecture



Intra-server Communication Simulator: Executes the analytical model, with different empirical parameters according to the operator type and GPU type.

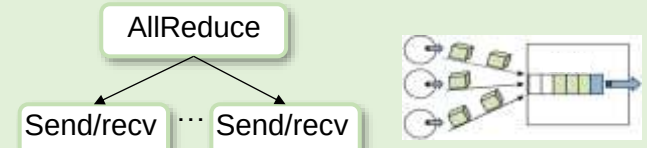
System simulator

Intra-server communication simulator (Analytical model)

AllReduce
AllToAll
...
AllGather

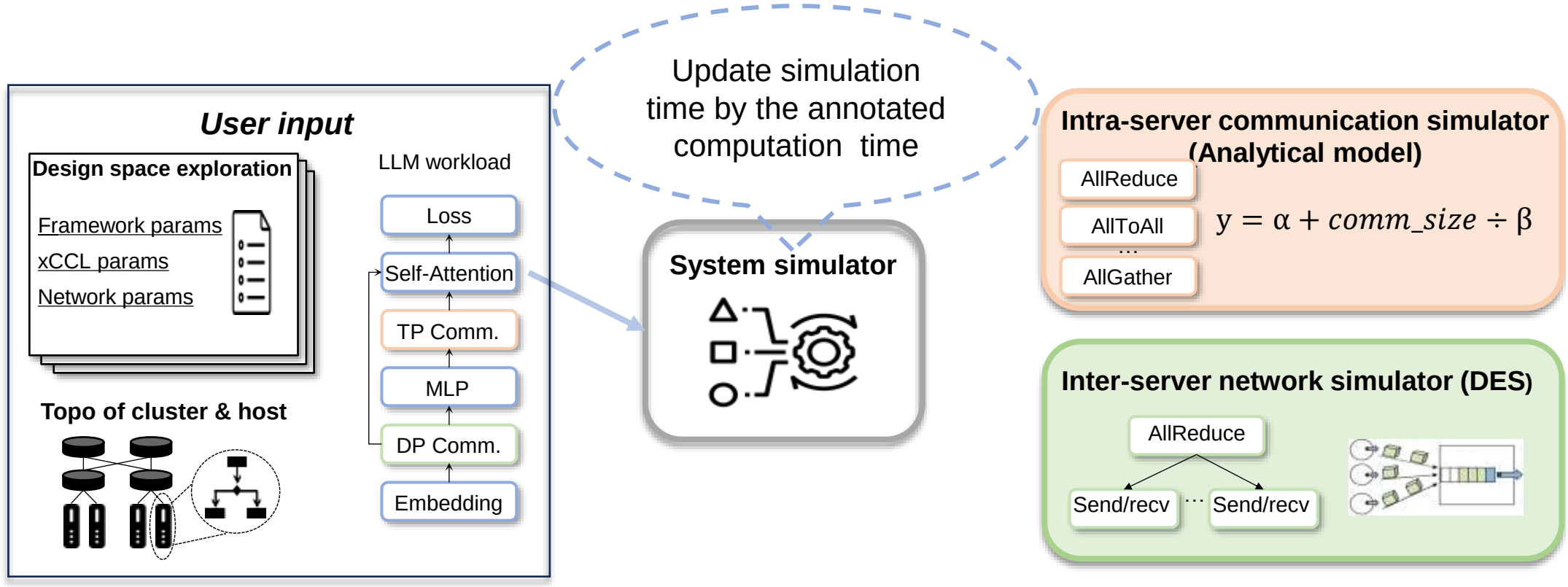
$$y = \alpha + comm\_size \div \beta$$

Inter-server network simulator (DES)

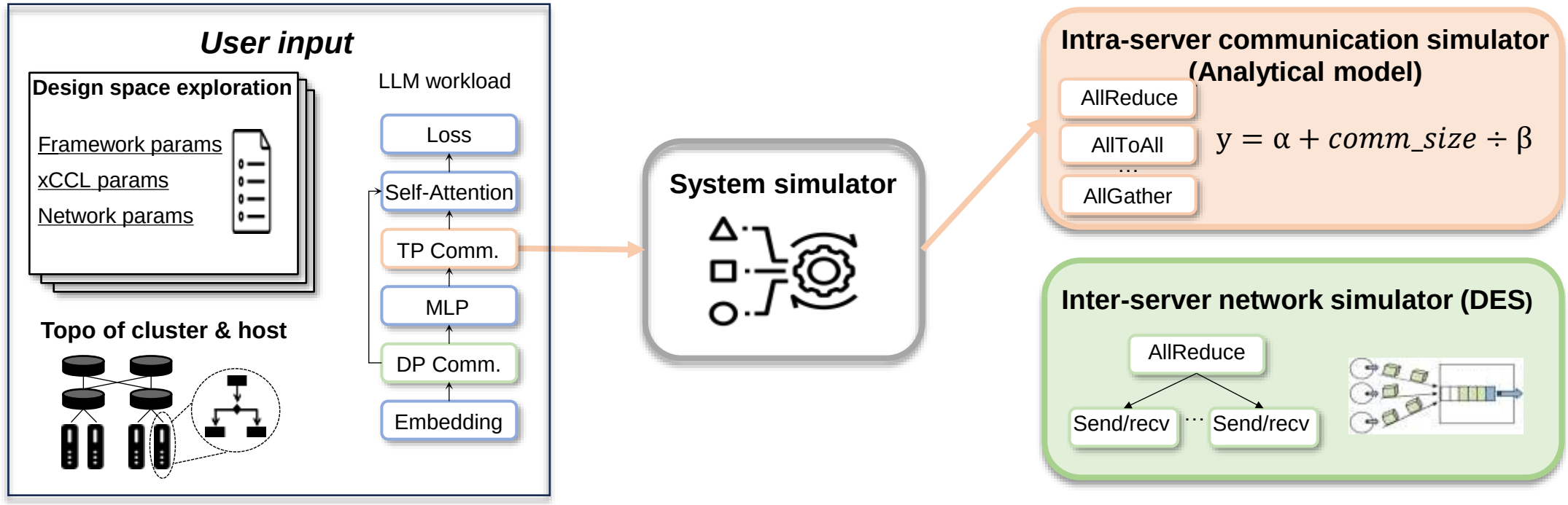


Inter-server Network Simulator: Conducts a packet level discrete-event simulation (DES), given point-to-point cross-network communication demands.

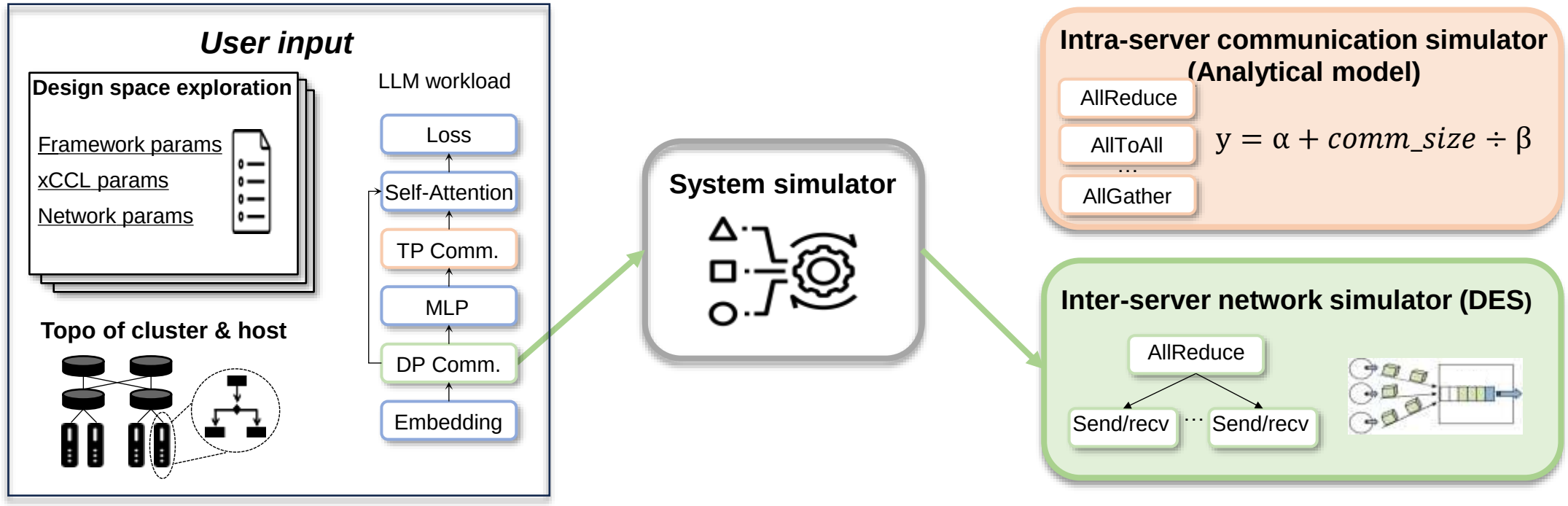
Multiverse workflow



Multiverse workflow



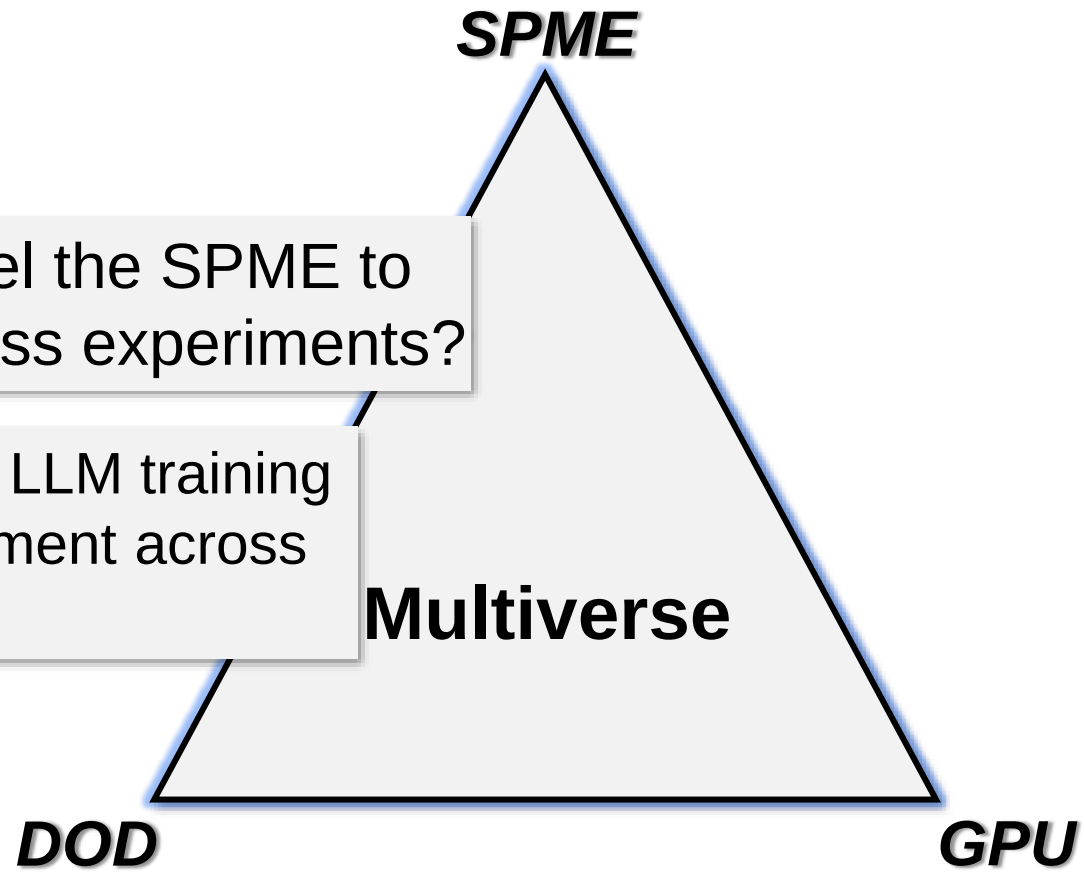
Multiverse workflow



Technique challenges

Challenge #1: how to model the SPME to exploit the parallelism across experiments?

Design #1: DOD modeling for LLM training and Column-storage management across experiments

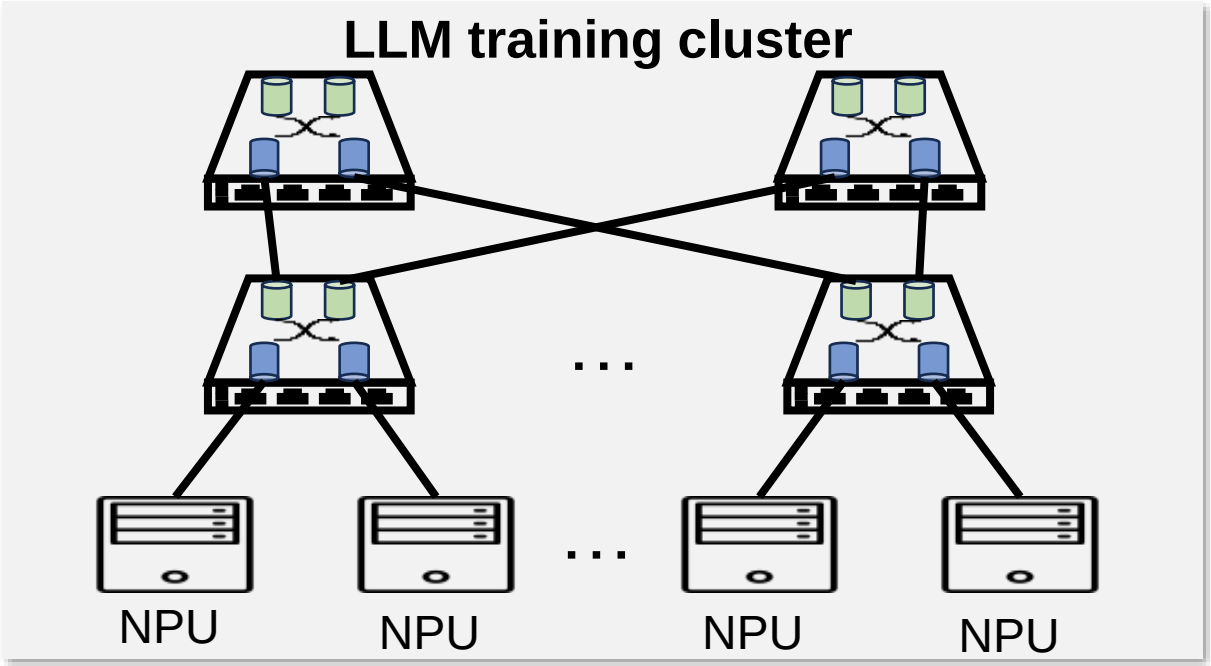


Design #1: Batch modeling for LLM training

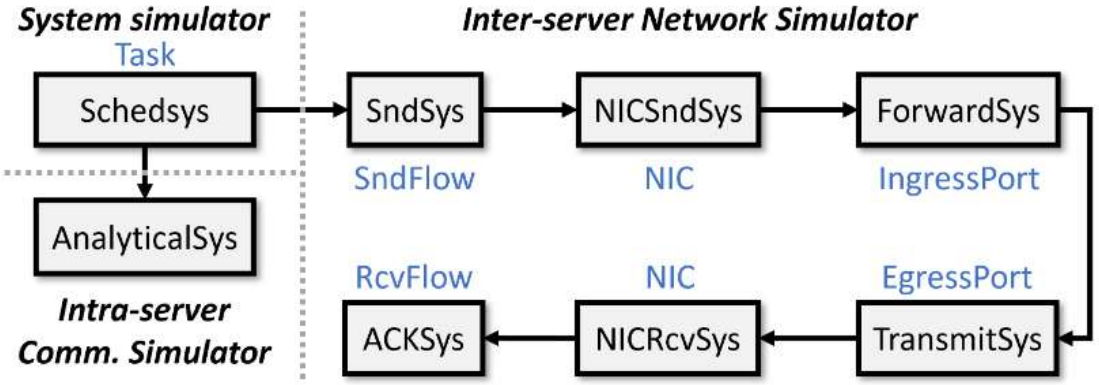
Entities	Data
Task	Type, Load, Pred_nodes, Succ_nodes, states
Flow	Stats, 5 tuples, Flow_size, CC_variable
Ingress Port	MAC, Buffer, FIB, Stats
Egress Port	Buffer, Stats， Schedule

Simulation logic

ScheduleSystem	Check predecessor status, activates successor **tasks**, advance sim. time, generate flow events for **senders**.
SendSystem	Generate *packets* in **Sender** and moving them to the connected **Ingress Port**
ForwardSystem	forward the *packets* in the **IngressPort** to the corresponding **EgressPort**
TransmitSystem	transmit the *packets* in **EgressPort** to the corresponding **IngressPort** or **Receiver**
ACKSystem	process the *packets* received by the **Receiver** and triggering ACKs

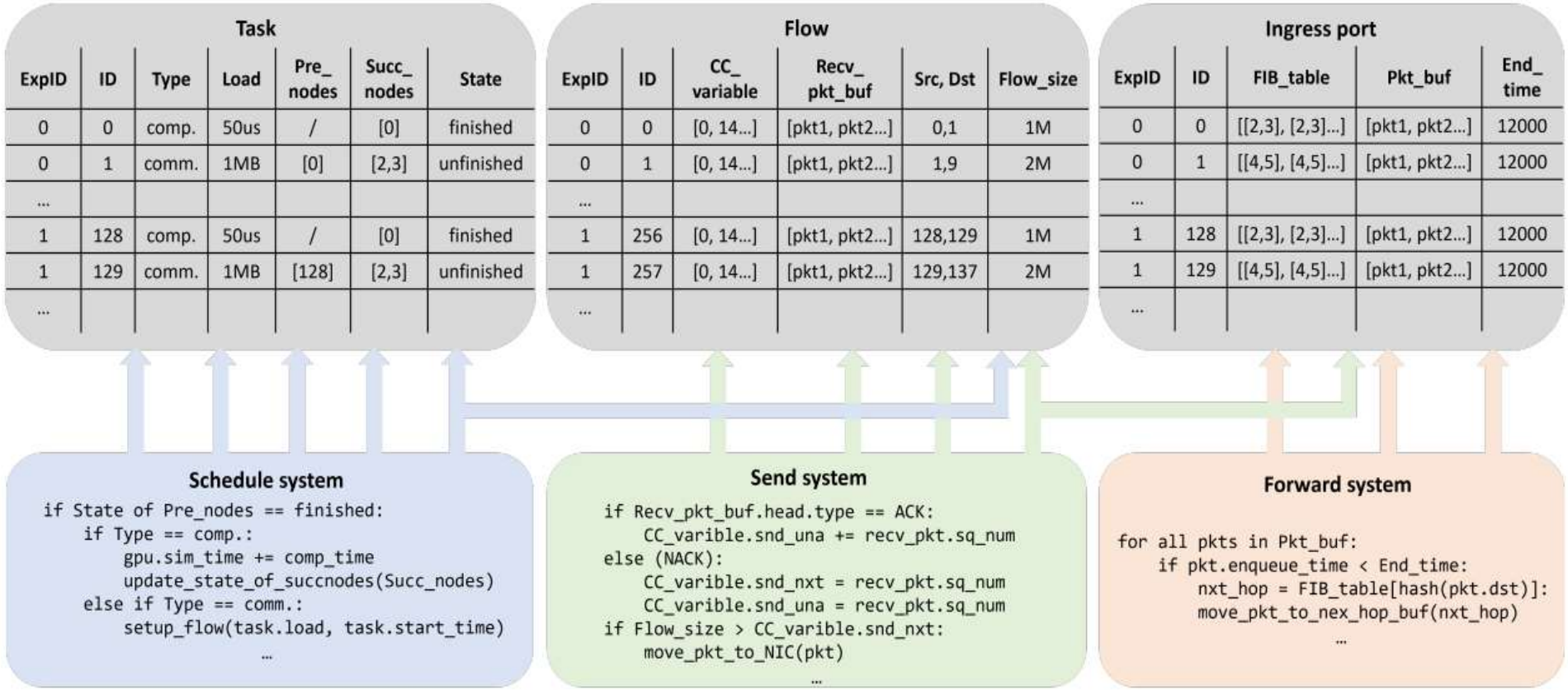


Simulation execution graph

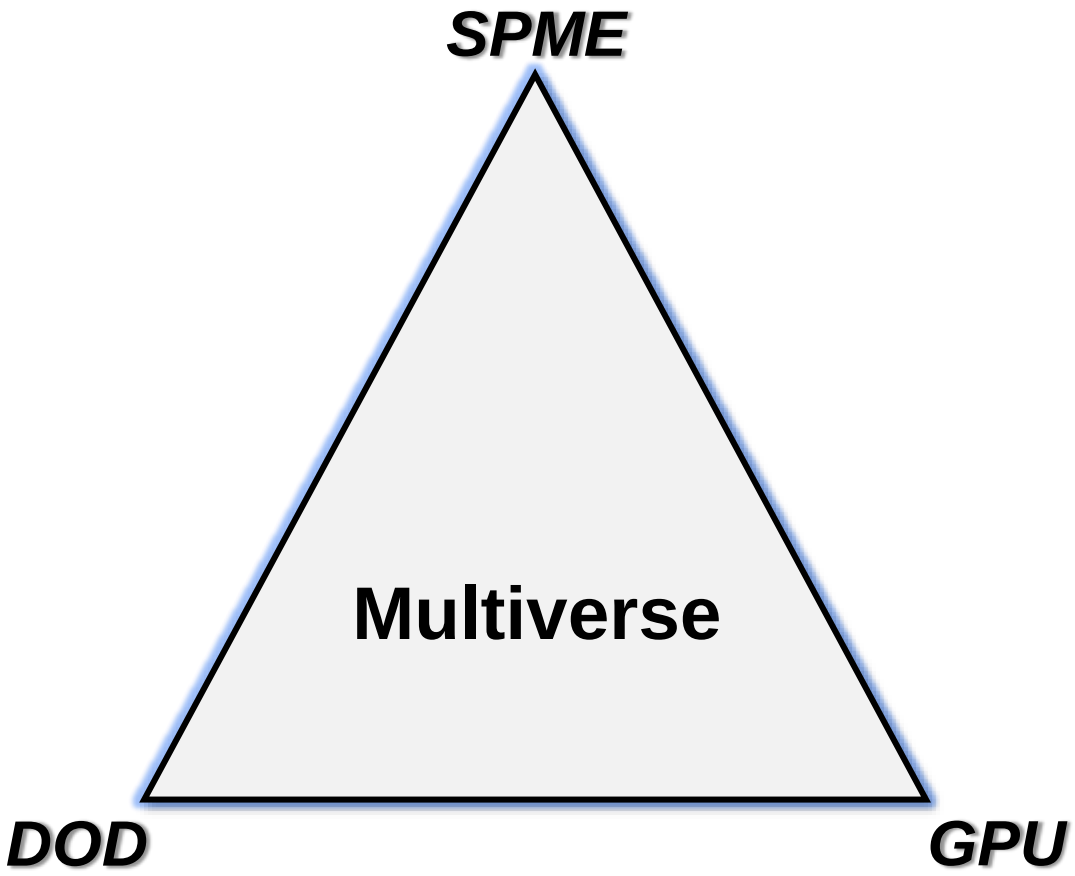


Design #1: Column-storage management across experiments

- A centralized table stores all data from multi-experiment.
- Data for the same kind of entity is stored contiguously in memory.



Technique challenges



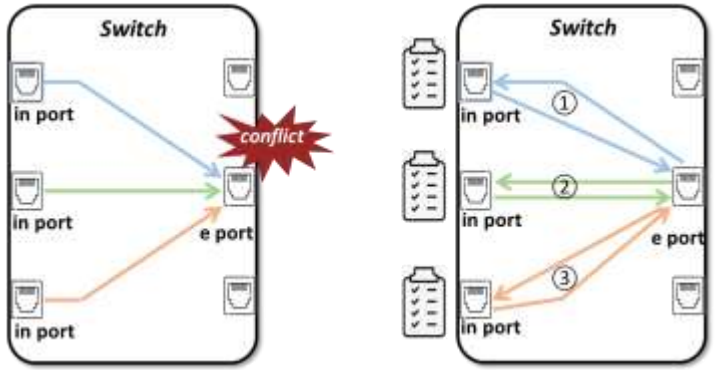
Challenge #2: how to execute DOD efficiently in GPU?

Design #2: Pull-based and kernel fusion technique

Design #2: Pull-based and kernel fusion technique to reduce the cost of synchronization

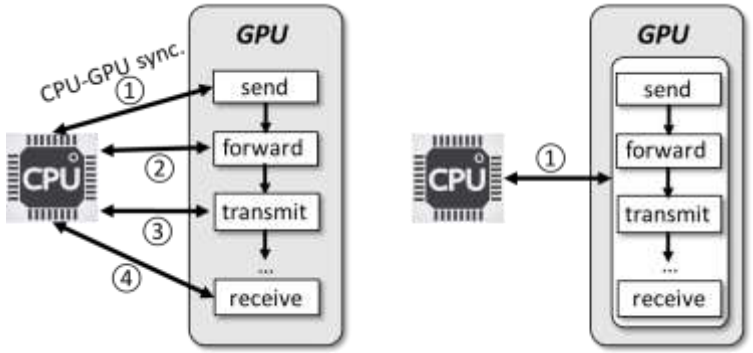
1. The lock-based synchronization overhead is very high.
2. The synchronization overhead between GPU and CPU is high.

1 Pull-based synchronization



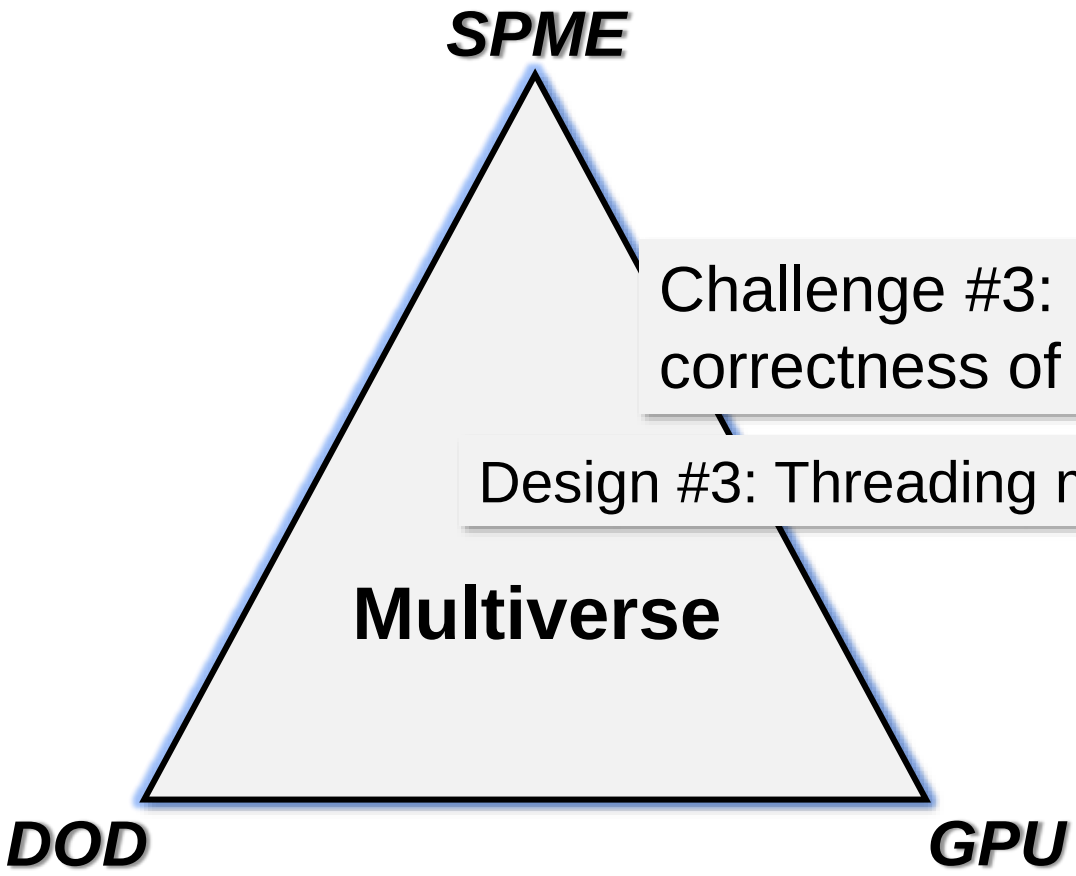
- Resolves the many-to-one write conflict.
- Enables lock-free multi-threading.

2 Kernel-fusion technique



- Fusing systems together reduces the total occurrence and duration of CPU-GPU communications.
- Also greatly increases GPU utilization.

Technique challenges

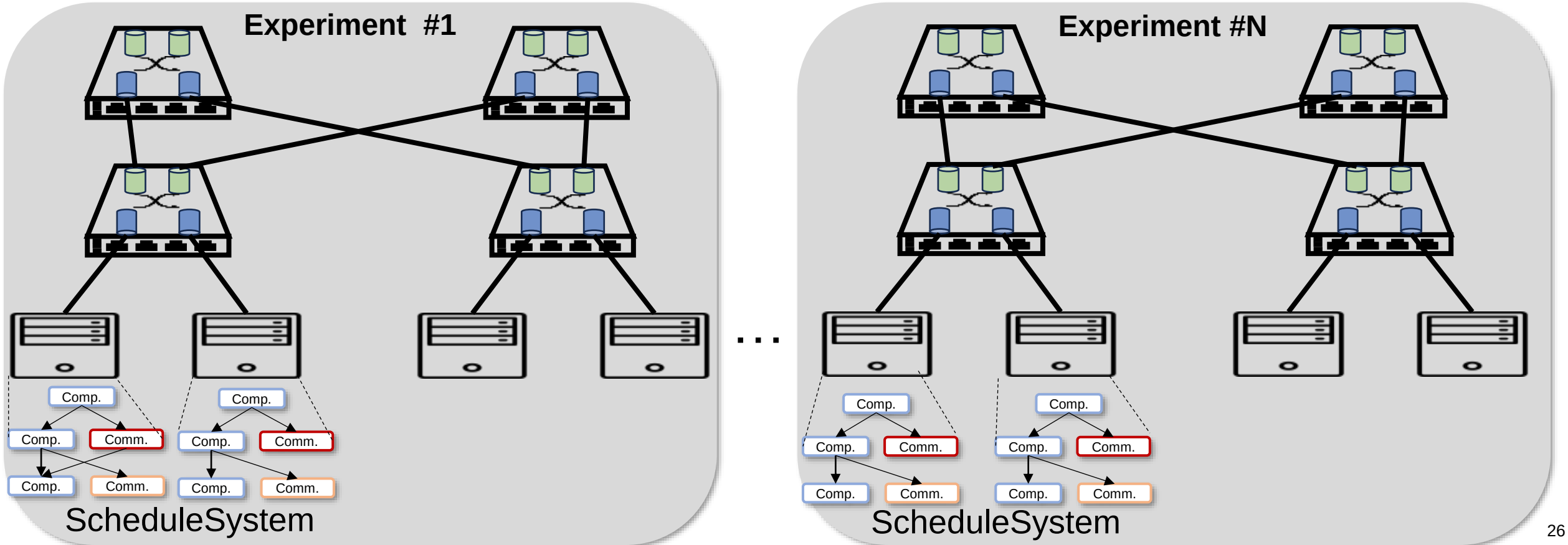


Challenge #3: How to guarantee the correctness of the execution order?

Design #3: Threading model across experiments

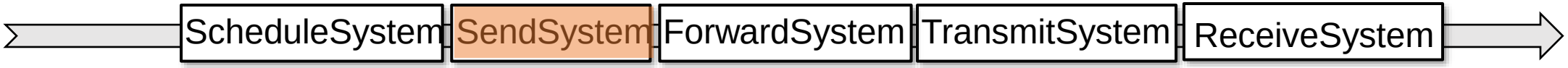
Design #3: Threading model across experiments

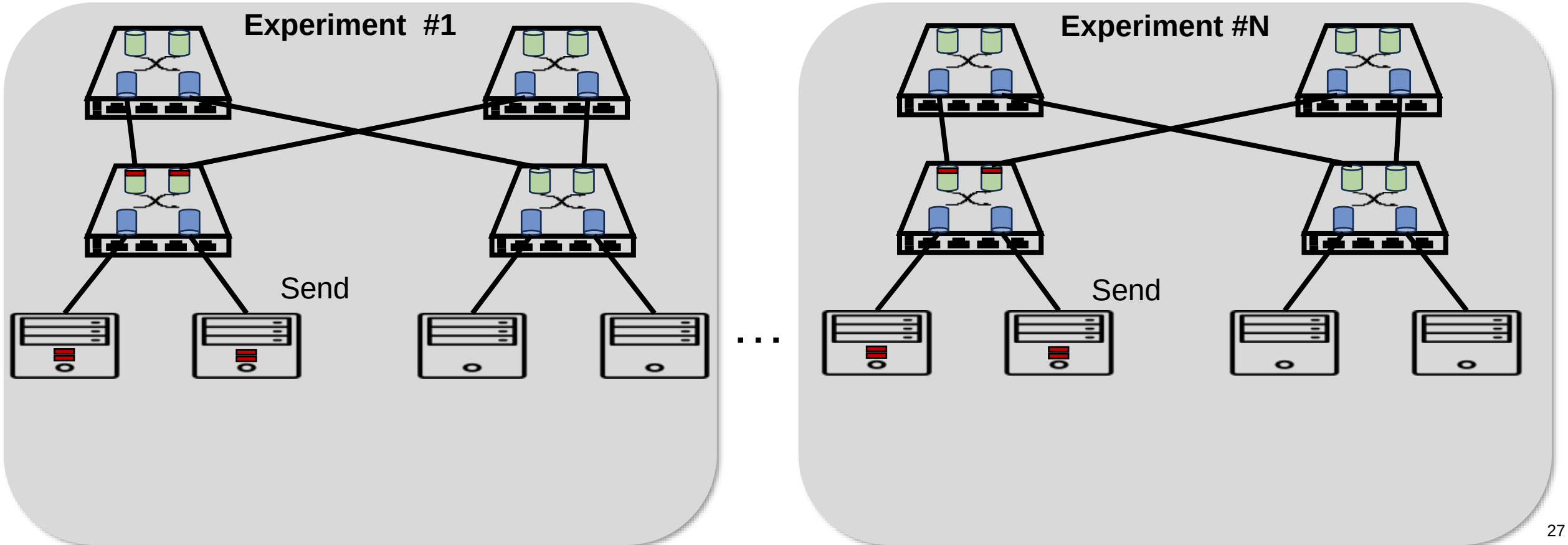
- Sequential execution for different systems in the simulation execution graph
- Batch and parallel execution for the same system of all experiments on GPU Cores



Design #3: Threading model across experiments

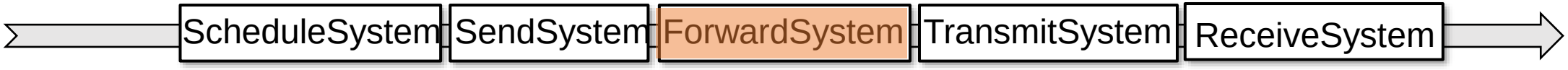
- Sequential execution for different systems in the simulation execution graph
- Batch and parallel execution for the same system of all experiments on GPU Cores

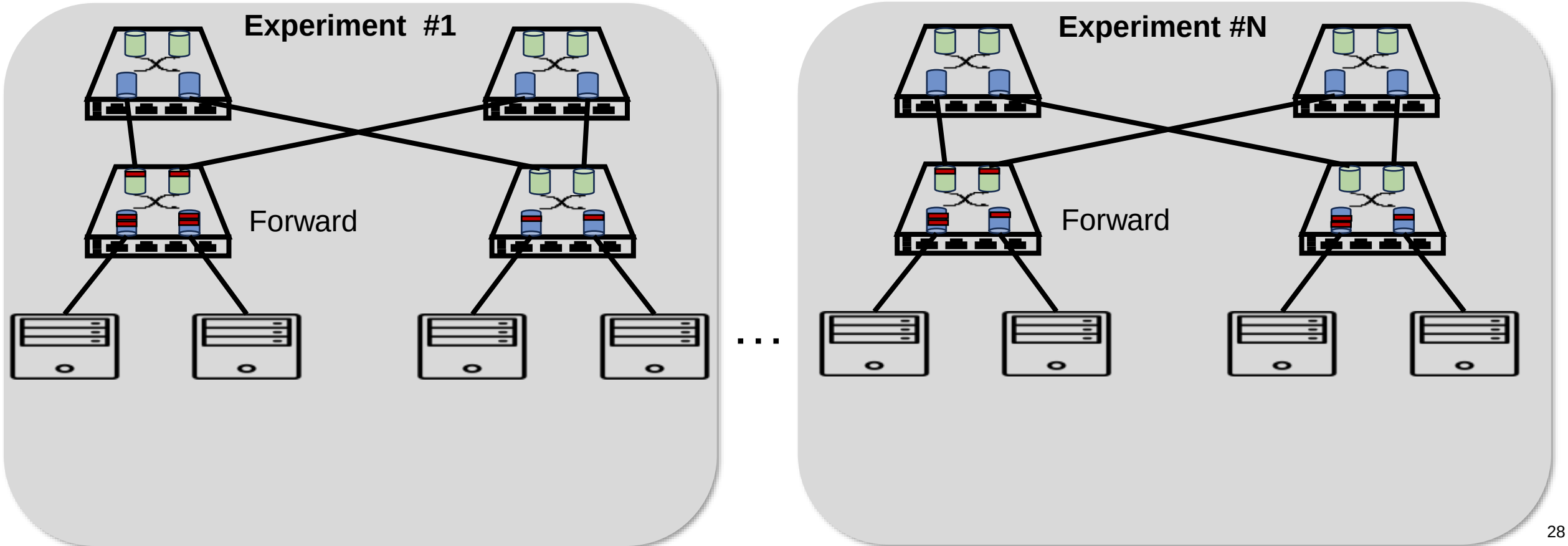




Design #3: Threading model across experiments

- Sequential execution for different systems in the simulation execution graph
- Batch and parallel execution for the same system of all experiments on GPU Cores





Implementation

- Programming language: C++20
- In addition to **GPU** execution, Multiverse also supports **CPU** execution.
- Features

Feature category	Options
Parallel strategies	TP, DP, PP
Collective communication algorithms	Ring/Tree allreduce, allgather, reducescatter
Topology	Fattree, Bcube, HPN, User-defined topo
Congestion control algorithms	DCQCN, DCTCP

- Actively **evolving** project
  - Multiverse 1.0 and 2.0 is open-sourced

https://github.com/NASP-THU/multiverse

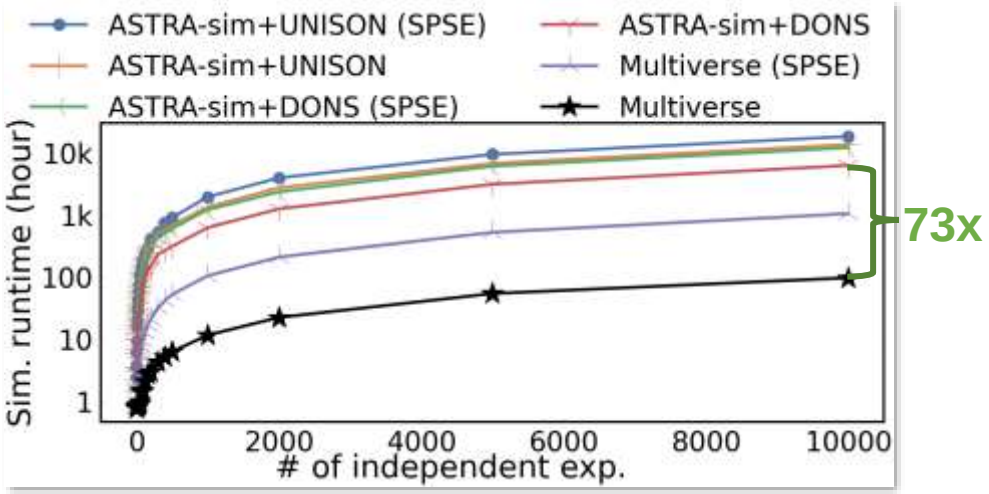
Evaluation setting

- Question： **Simulation speed? Scalability? Accuracy?**
- Comparison：ASTRA-sim+UNISON, ASTRA-sim+DONS, Multiverse (SPSE).
- Setup: a server with one NVIDIA H100 GPU, an 80-core Intel CPU.

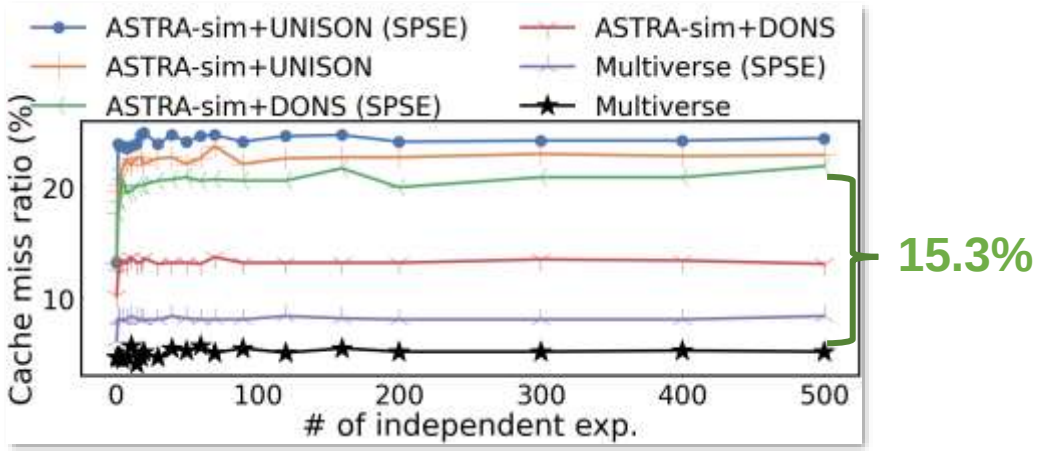
Use case	GPU scale	Topology	Workload	Explored parameters	# of independent exp.
#1: Topology optimization	128 GPUs	Fattree-like	GPT-3 13B	Set different layers, switch radix, connections, *etc.*	10k
#2: Collective communication optimization	1,024 GPUs	Fattree k=16	LLaMA 65B	Set different flow priority and load balancing strategies.	500
#3: Selection of TP/DP/PP group size	8,192 GPUs	Fattree k=32	GPT-3 175B	Set different TP/DP/PP group size.	100
#4: Comparing congestion control algorithms	54,000 GPUs	Fattree k=60	GPT-dense 175B	Set different CC algorithm, such as DCQCN, HPCC.	4

Multiverse is fast

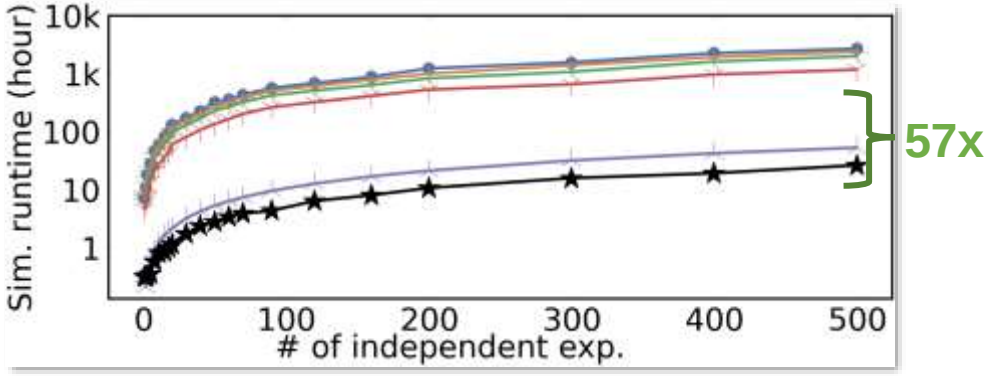
- Compared to the SOTA AI training simulators, Multiverse improves simulation speed by **57x to 73x**.
- In single experiment, Multiverse supports a maximum network scale of **54k GPUs**, achieving an acceleration ratio of **43x**.



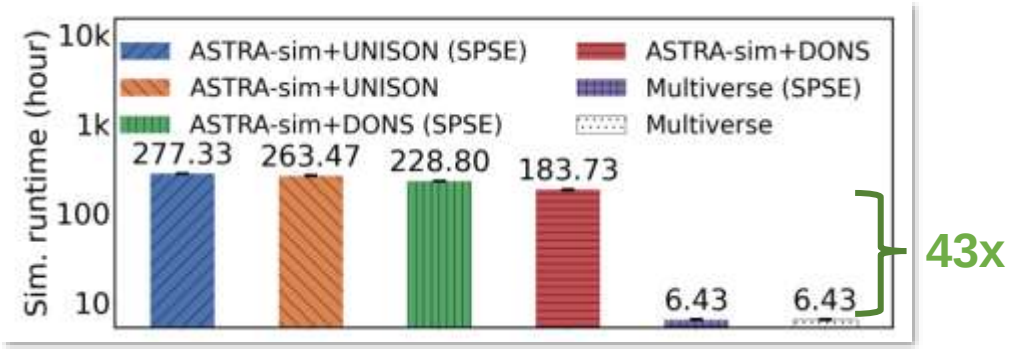
Case: Topo. optimization: 128 GPUs



Cache miss rate comparisons



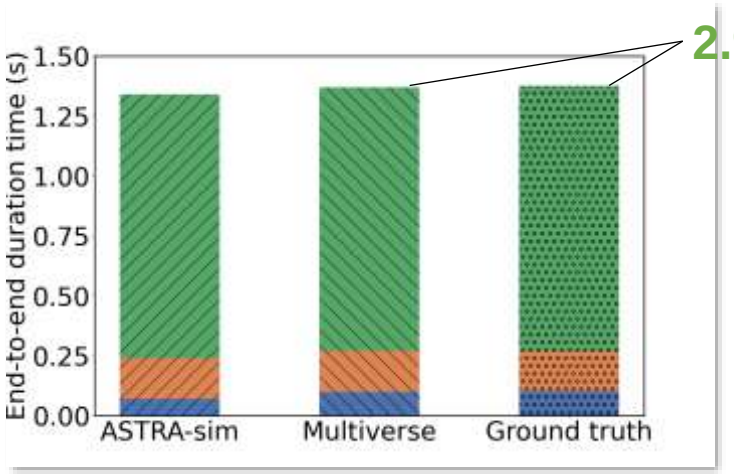
Case: Selection of TP/DP/PP group size: 8192 GPUs



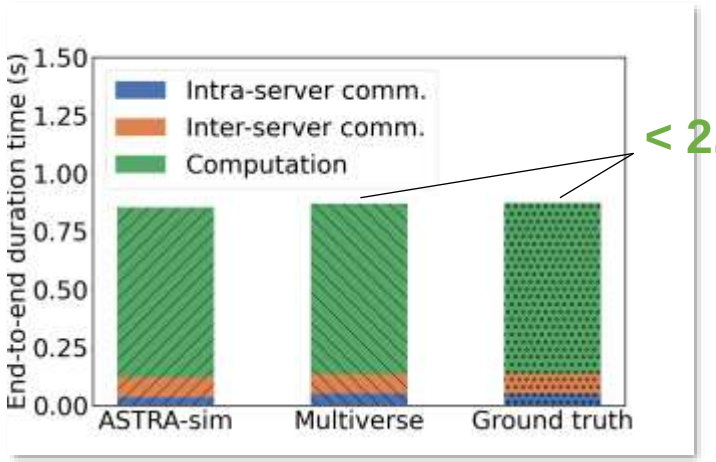
Simulating 54k GPUs

Multiverse is accurate

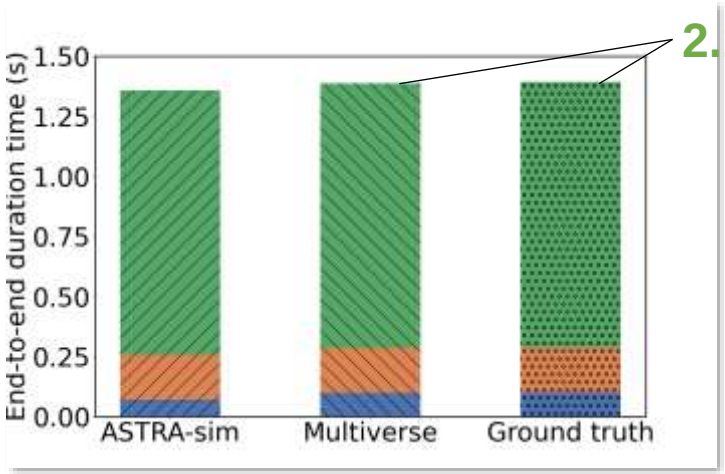
- Multiverse has an error rate of less than **3%** when compared to real LLM training data.



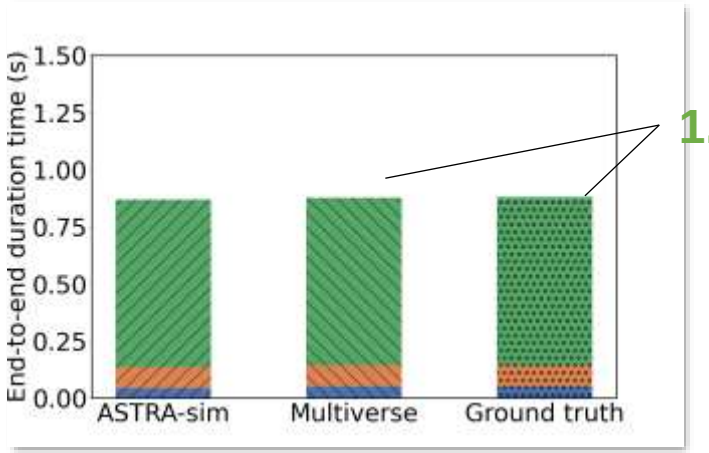
GPT3 175B, 128 GPUs



LLaMA 65B, 128 GPUs



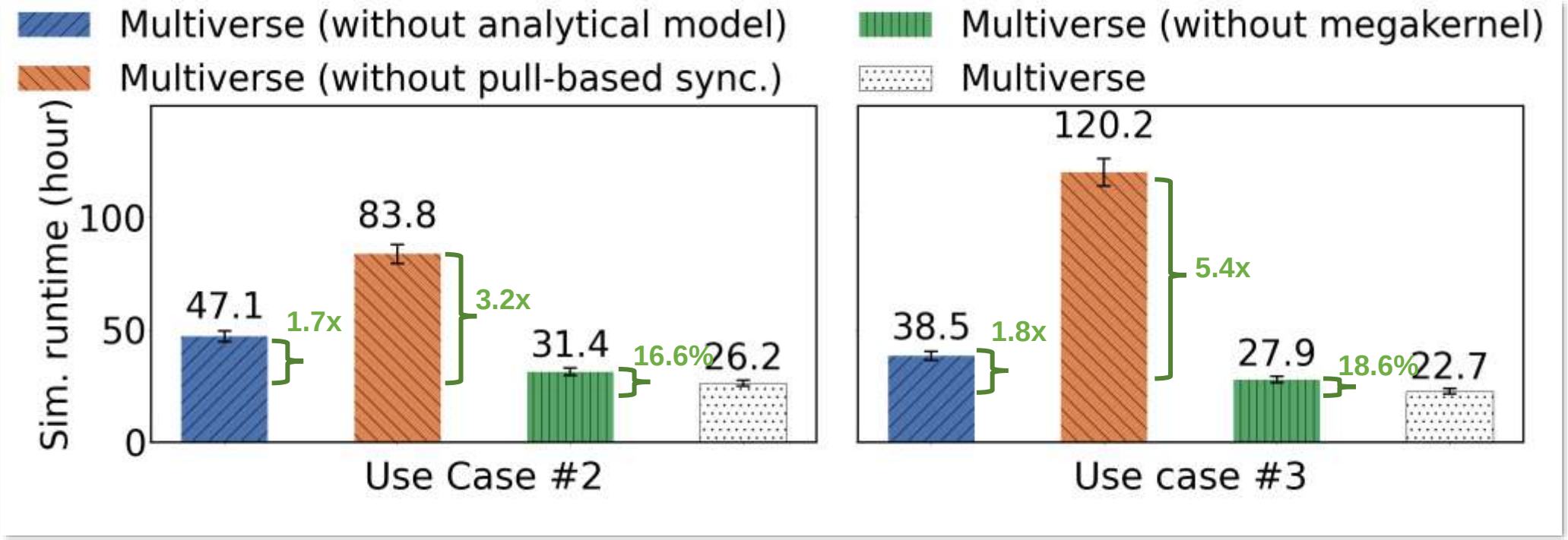
GPT3 175B, 1024 GPUs



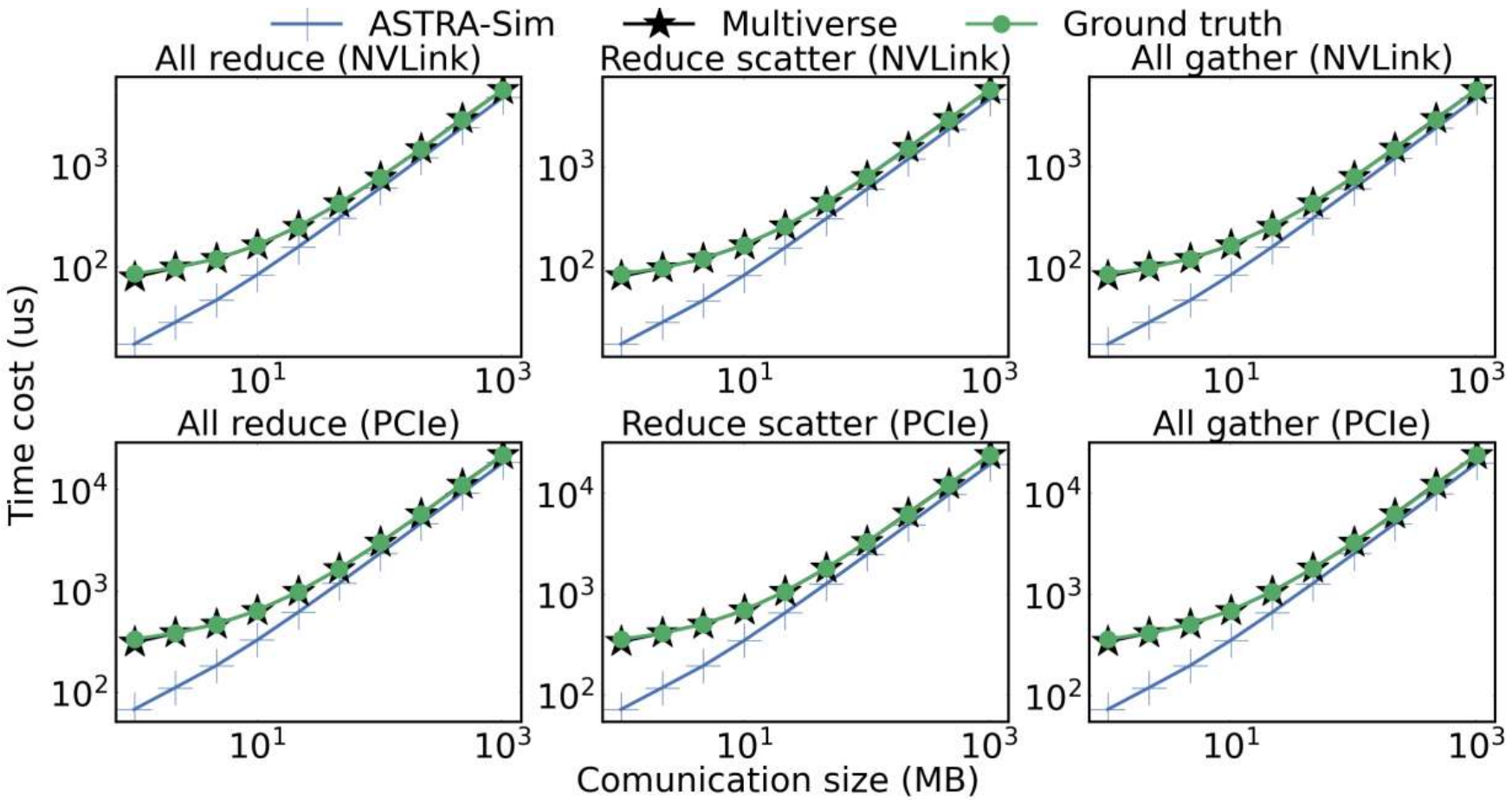
LLaMA 65B, 1024 GPUs

Ablation study

- Compared to Multiverse (without analytical model), Multiverse can achieve a speedup of **1.7-1.8×.**
- Compared to Multiverse (without pull-based synchronization), Multiverse can achieve a speedup of **3.2-5.4×**.
- Compared to Multiverse (without Megakernel), Multiverse can reduce the simulation time by **16.6%-18.6%**.



Analytical model for intra-server communication is accurate



- Experiment setting: 8 A100 GPUs interconnected via 300GB/s NVLink and PCIe5.0.
- Evaluation results: the error is merely 0.8%−1.2%.

Conclusion

Future directions

- More device and protocols (UniBus, NPU, etc.)
- Precise simulation of intra- and inter- machine simulations (Unifying scale-up and scale-out simulation).
- Fault-tolerance simulation for LLM training systems.
- …

https://github.com/NASP-THU/multiverse

Thanks!