Phantora: Maximizing Code Reuse in Simulation-based Machine Learning System Performance Estimation

Jianxing Qin, Jingrong Chen, Xinhao Kong, Yongji Wu, Tianjun Yuan, Liang Luo, Zhaodong Wang, Ying Zhang, Tingjun Chen, Alvin R. Lebeck, Danyang Zhuo

Duke UNIVERSITY

Meta

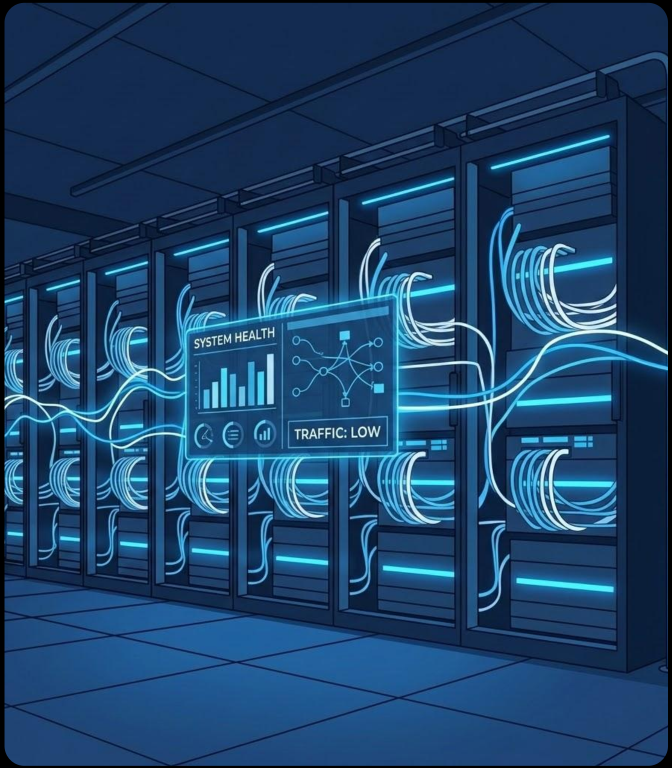
Performance Prediction is Critical

Economic Constraint

Real-world tuning costs enormous resources

Resource Planning

Provisioning decisions before hardware availability



Performance Modelling is Difficult

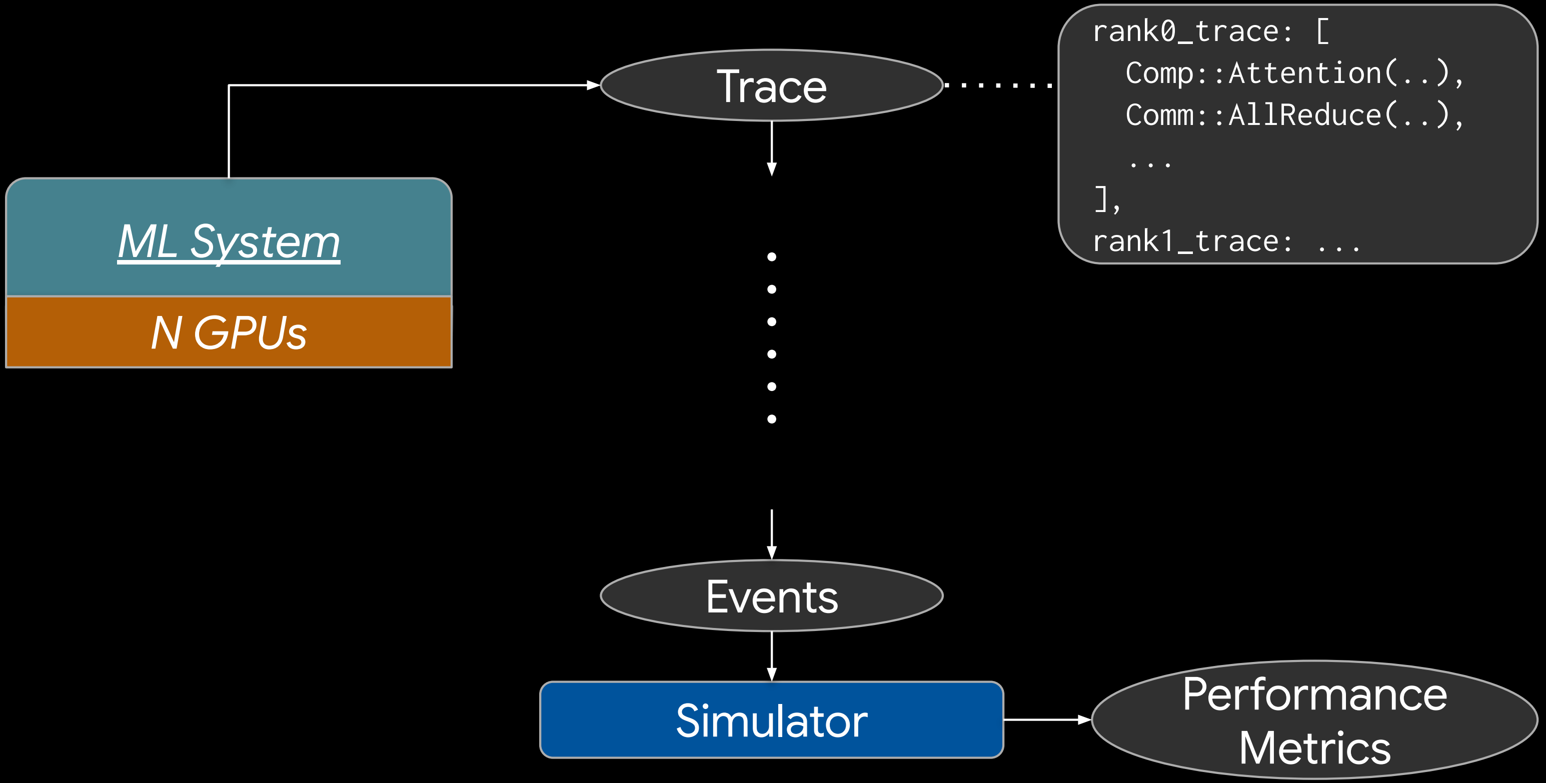
Complex configuration space

- Batch size
- Parallelization

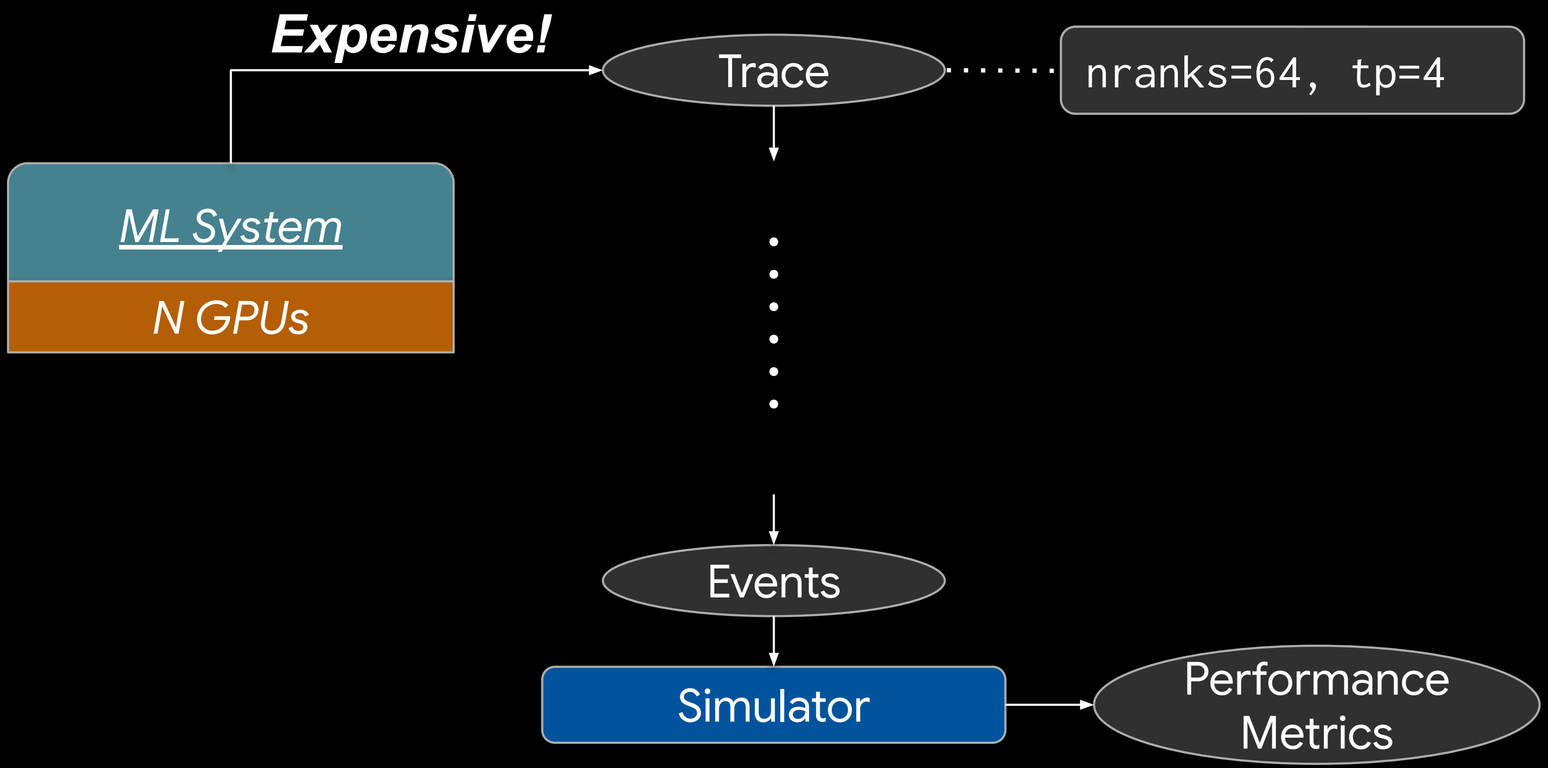
Framework-specific features

e.g., Selective activation recomputation

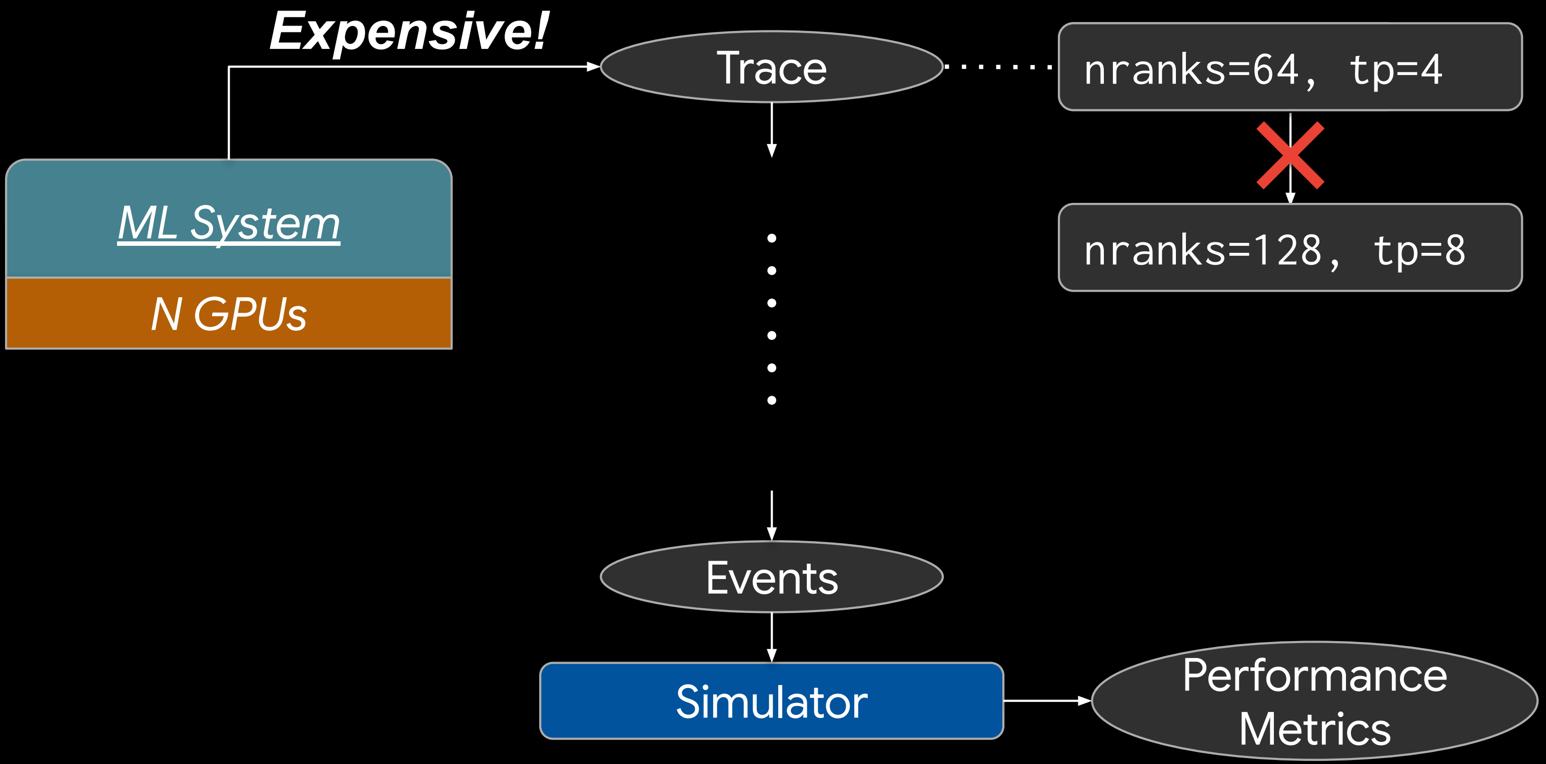
Trace-based Simulation



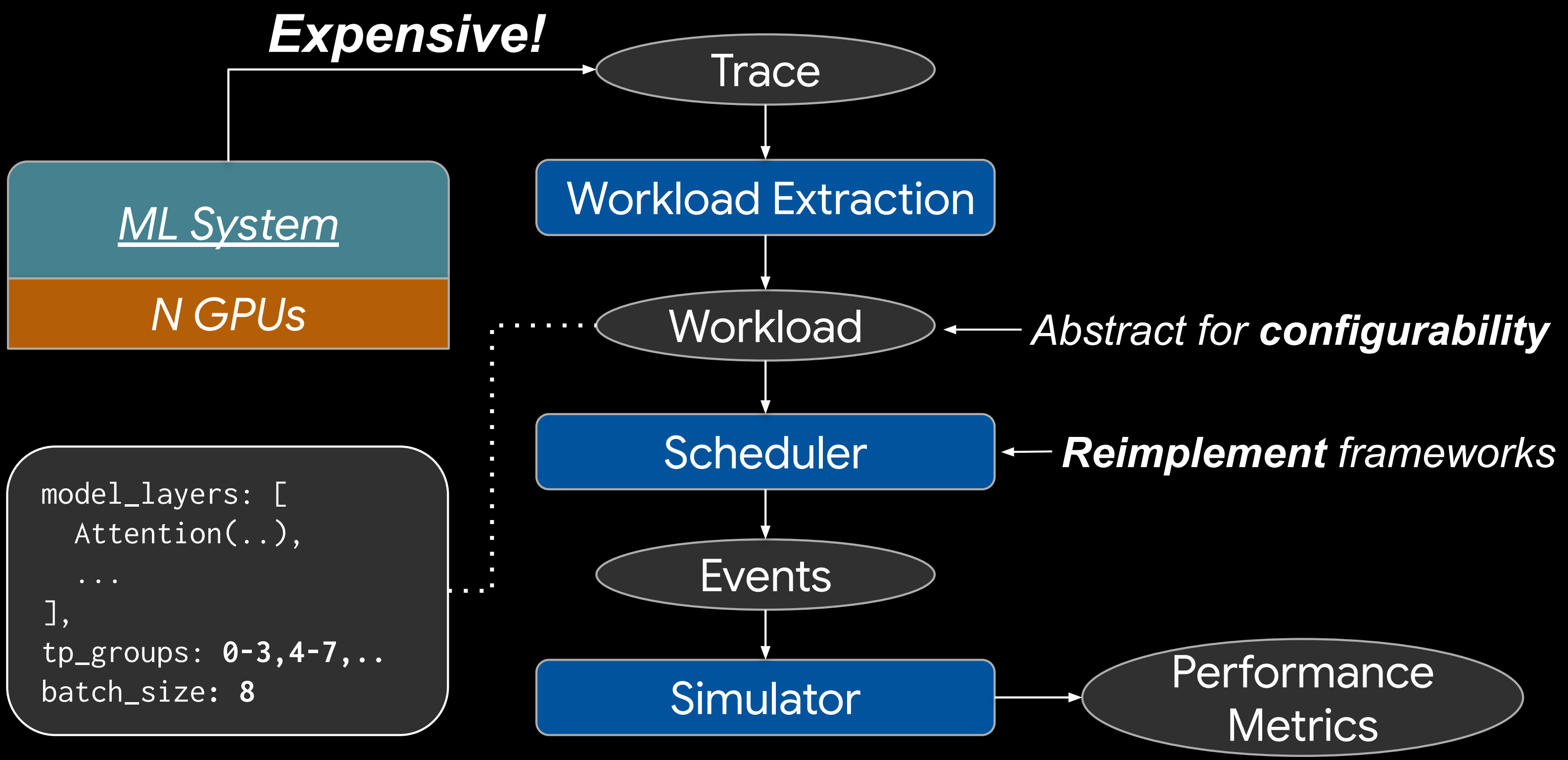
Trace-based Simulation



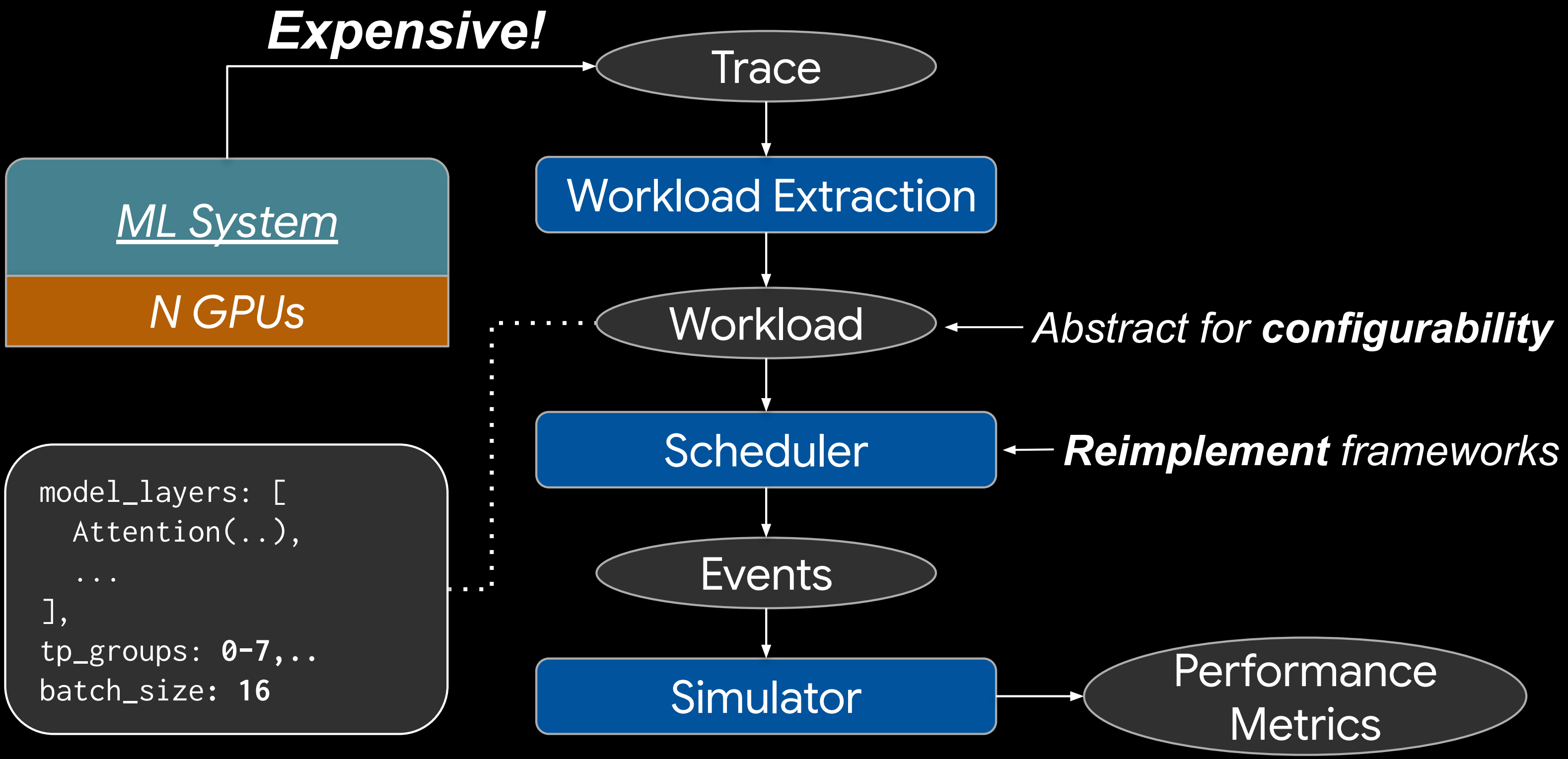
Trace-based Simulation



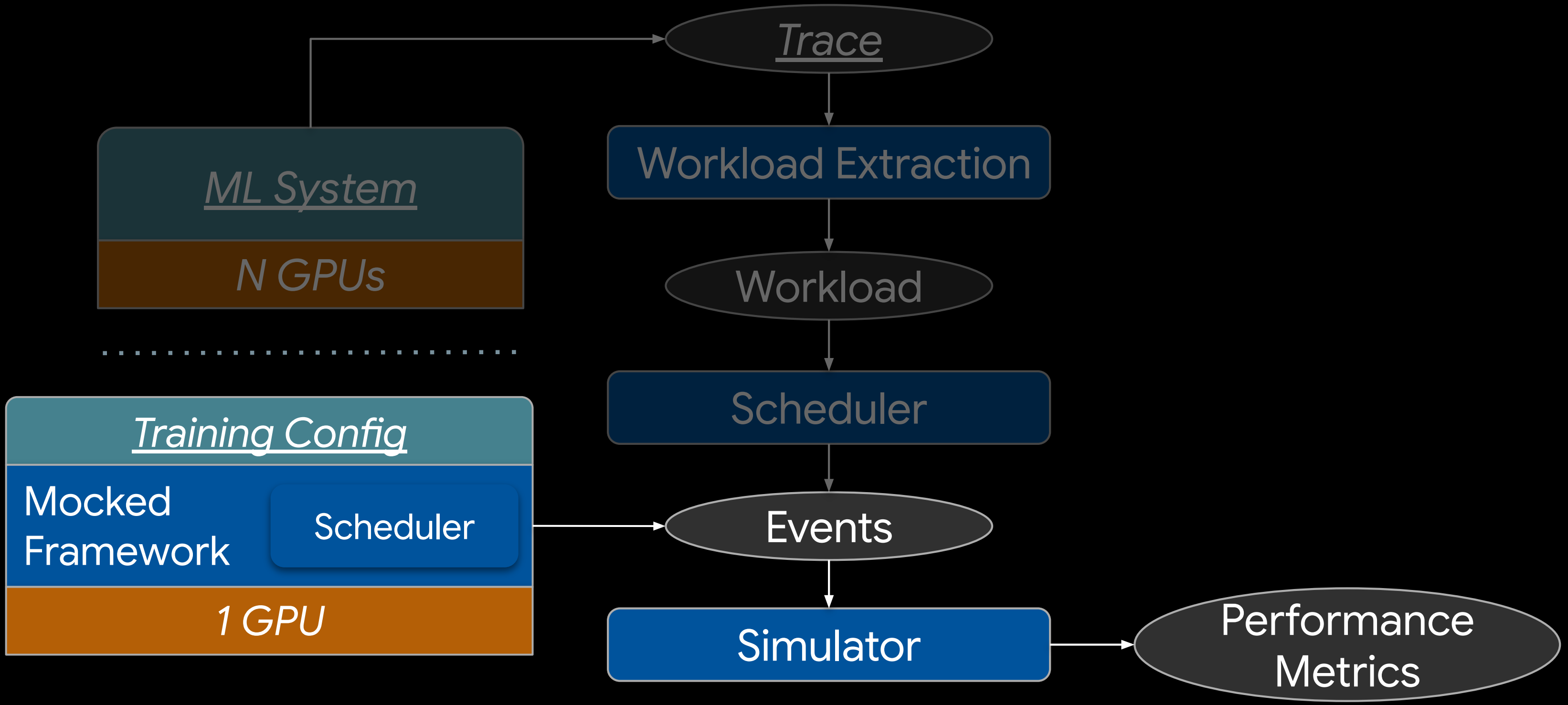
Reimplementation in Trace-based Simulation



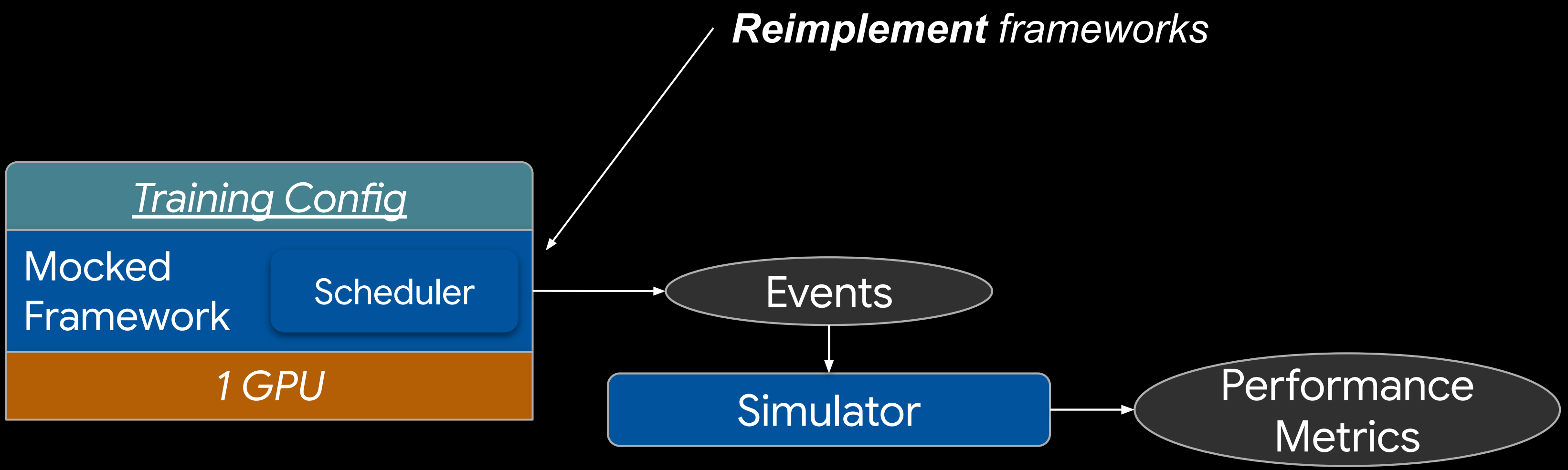
Reimplementation in Trace-based Simulation



Mocked Frameworks (SimAI, NSDI '25)



Reimplementation in Mocked Frameworks

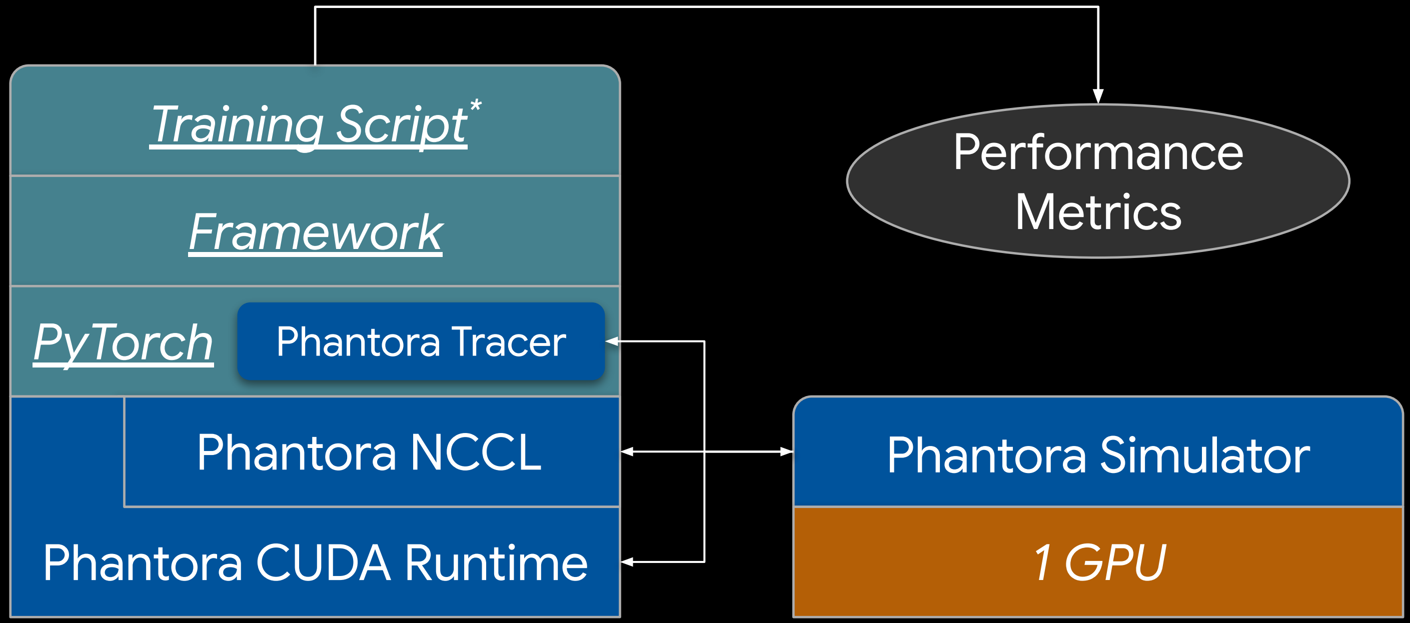


Can we directly ***reuse*** the framework's code for simulation?

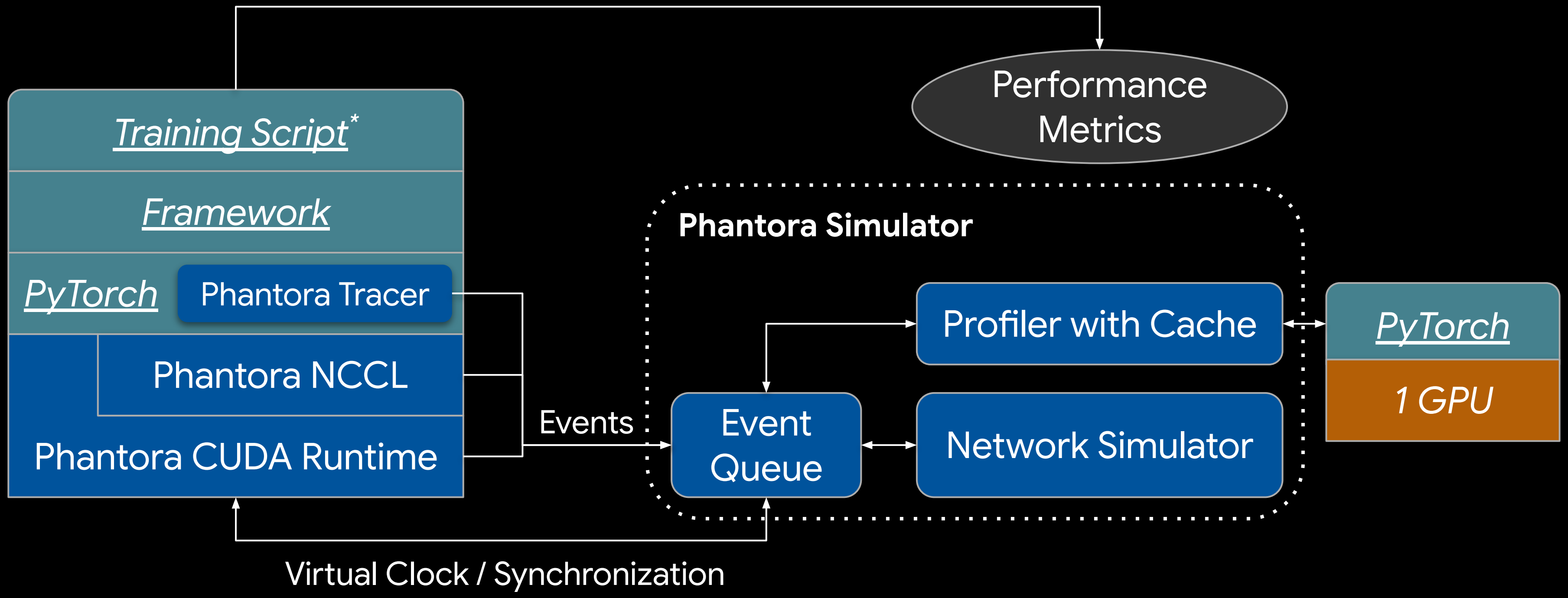
Can we directly ***reuse*** the framework’s code for simulation?

- Less maintenance effort
- Capture specific framework features

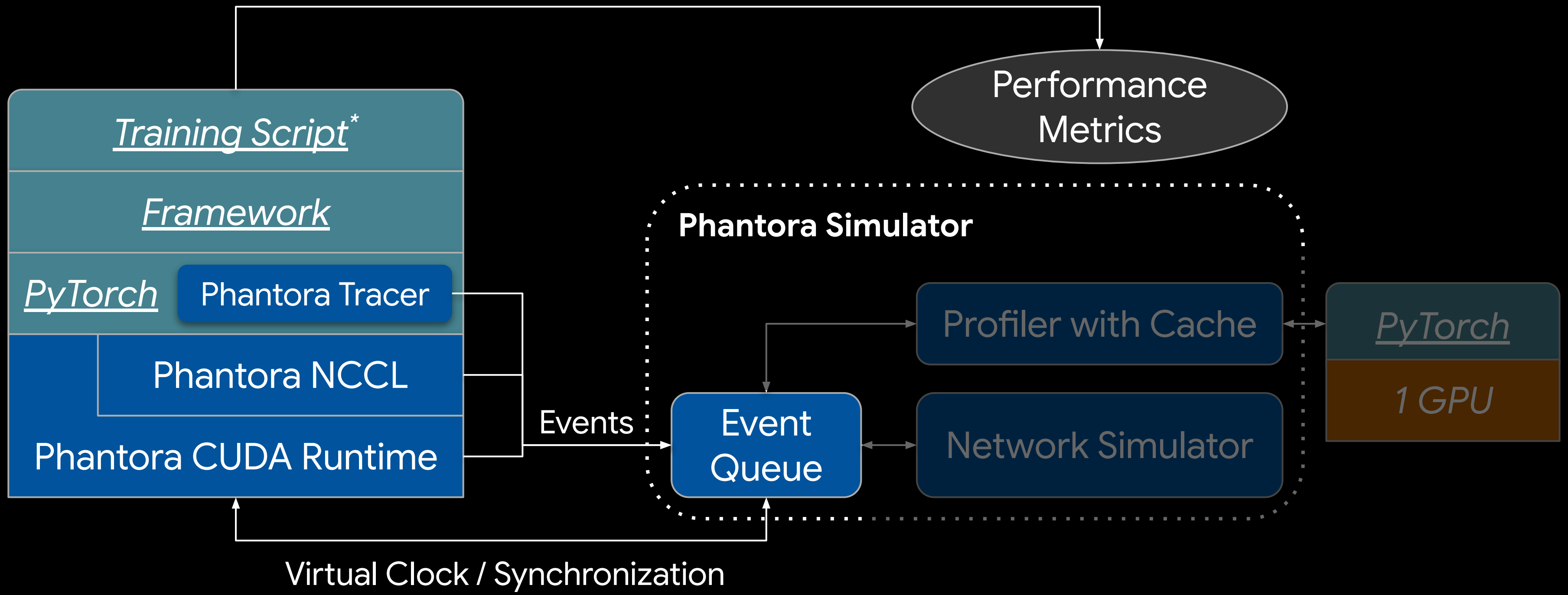
Phantora: Hybrid Simulation



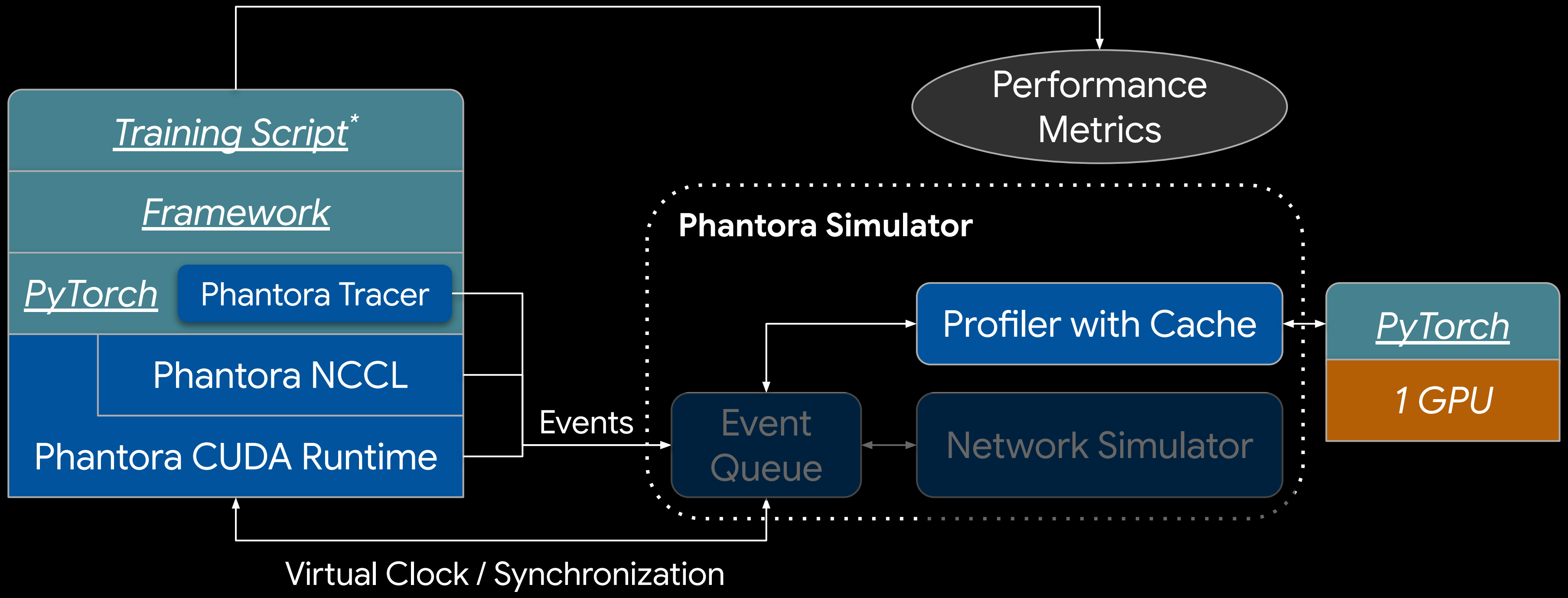
Phantora: Hybrid Simulation



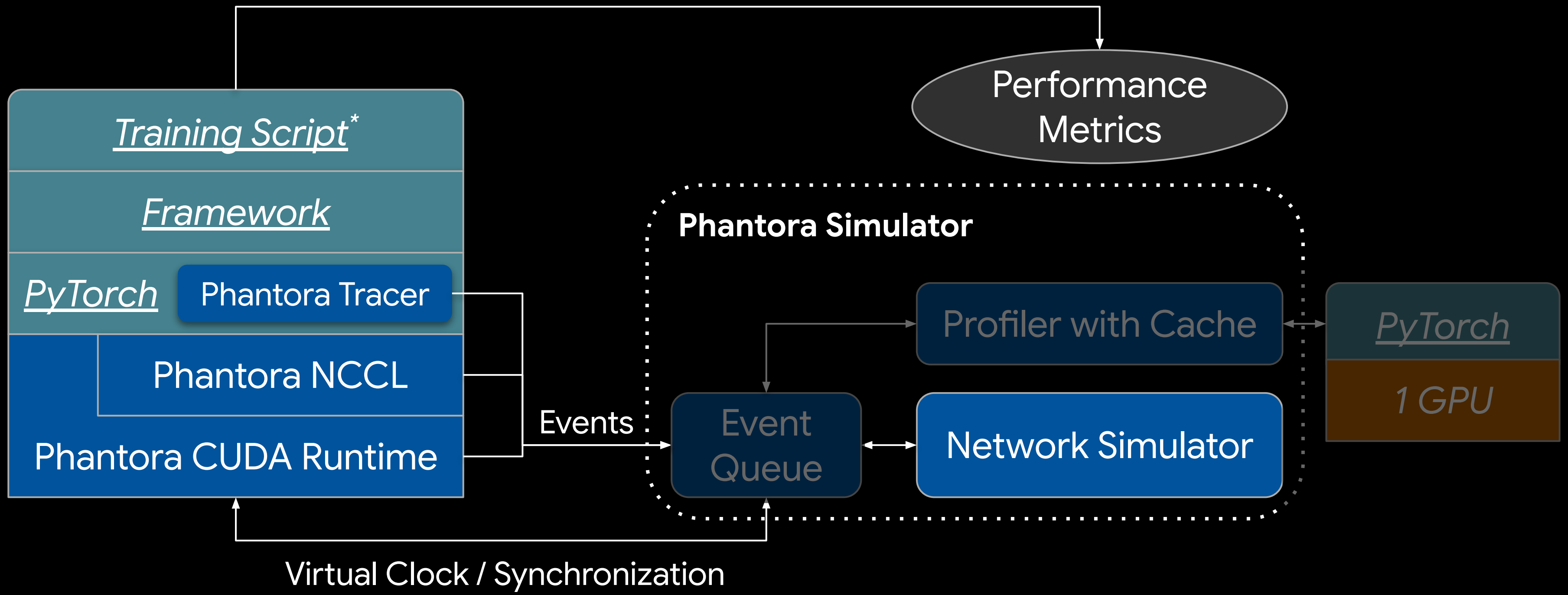
Phantora: Hybrid Simulation



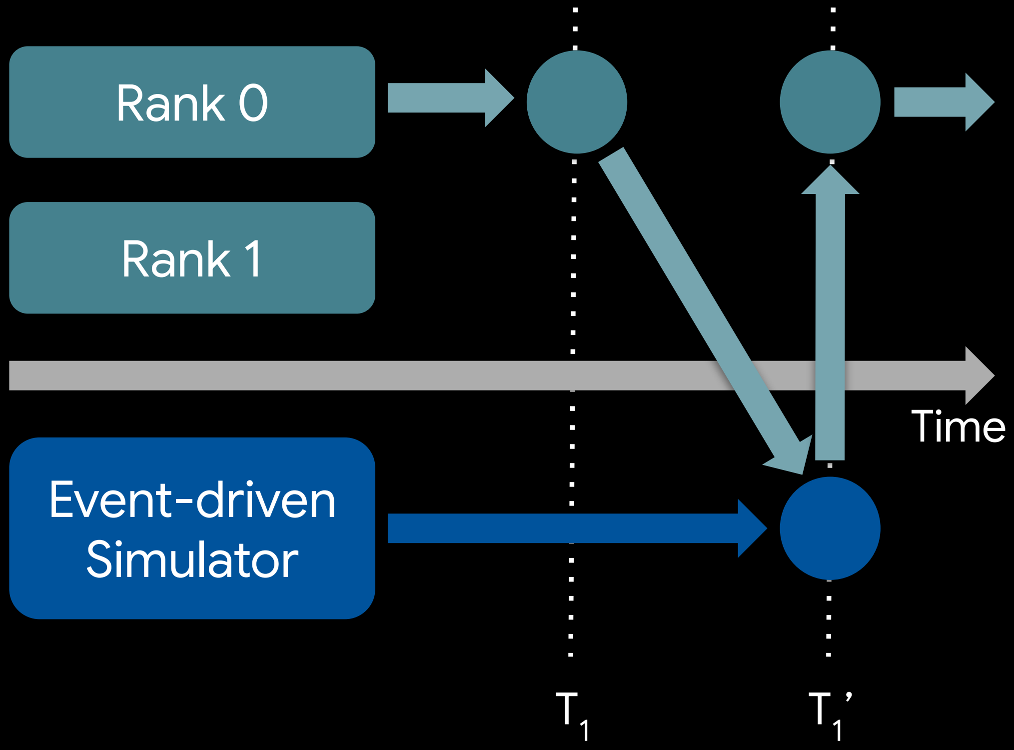
Phantora: Hybrid Simulation



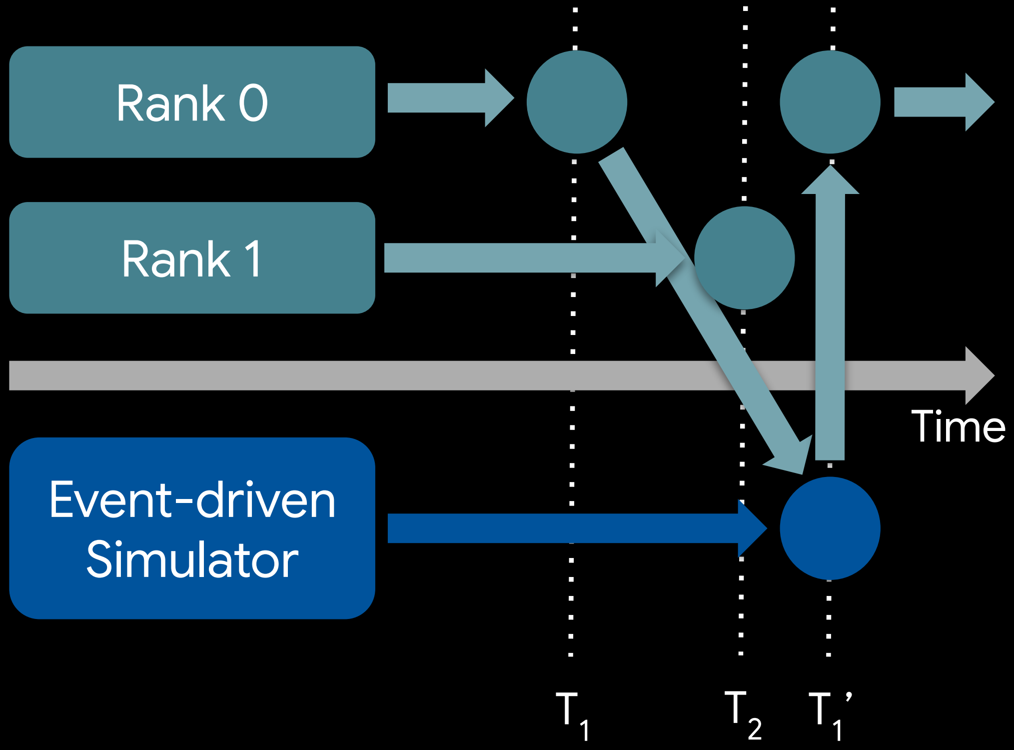
Phantora: Hybrid Simulation



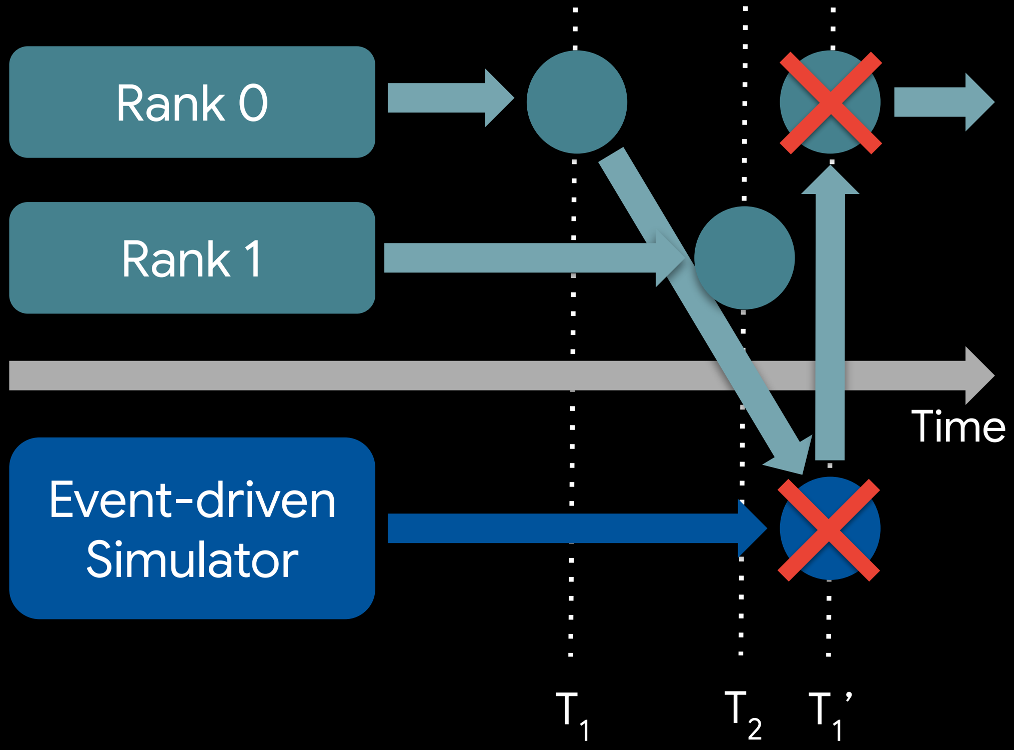
Problem: Past Events



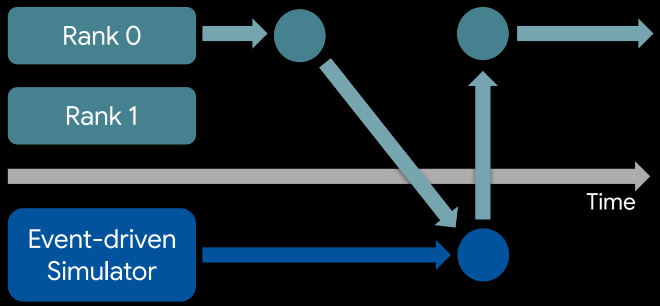
Problem: Past Events

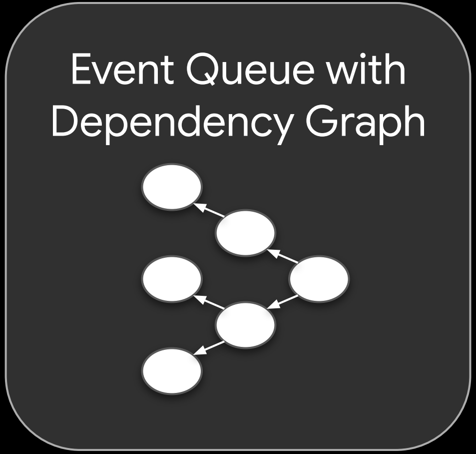


Problem: Past Events



Solution: Rollback



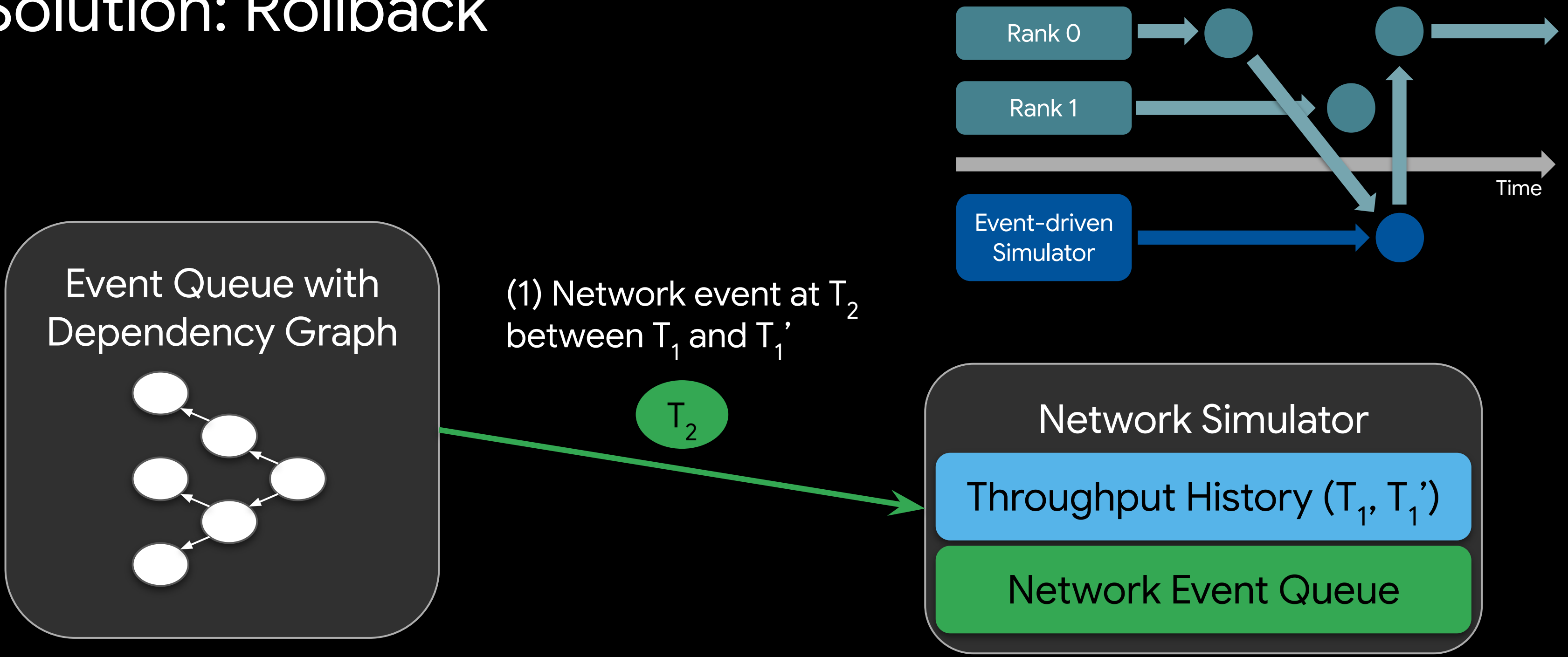


Network Simulator

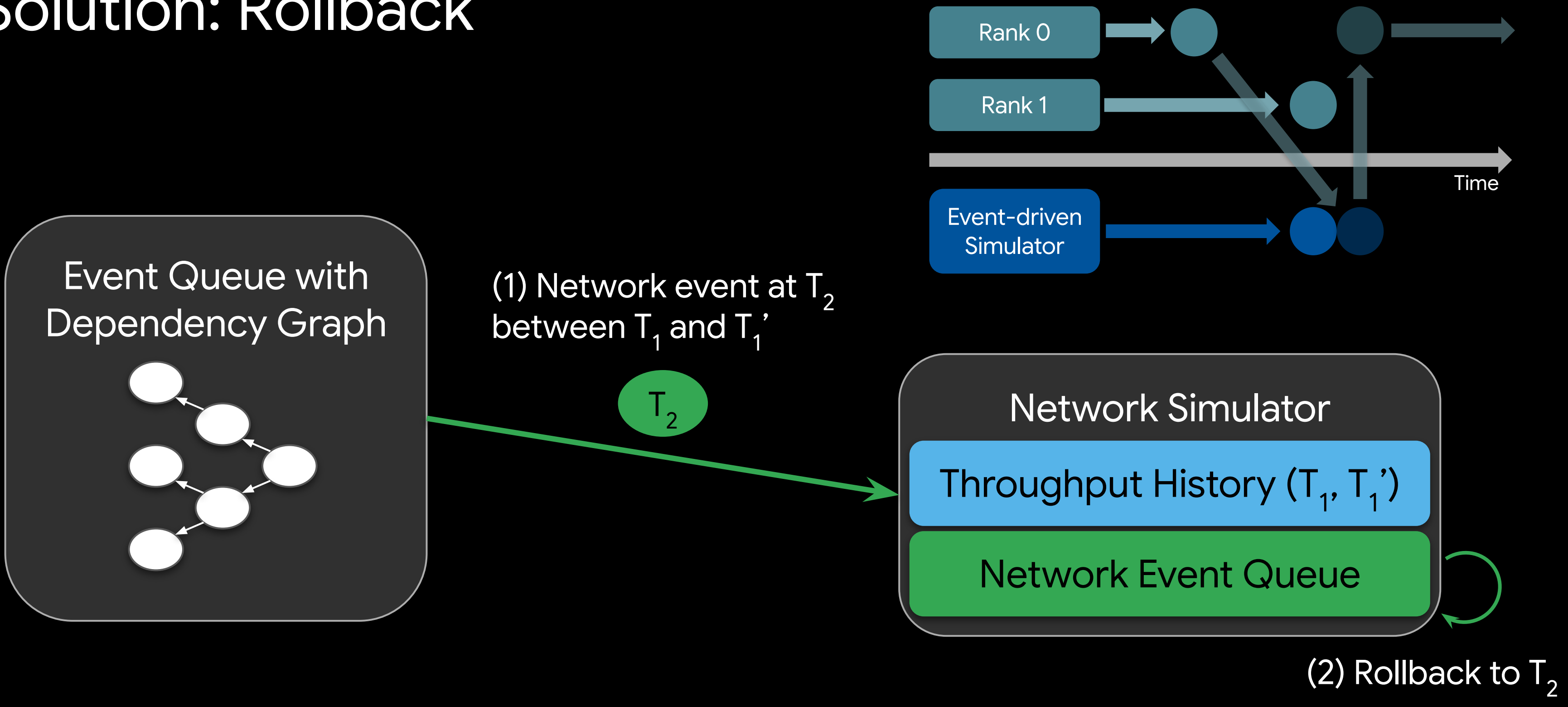
Throughput History (T_1, T_1')

Network Event Queue

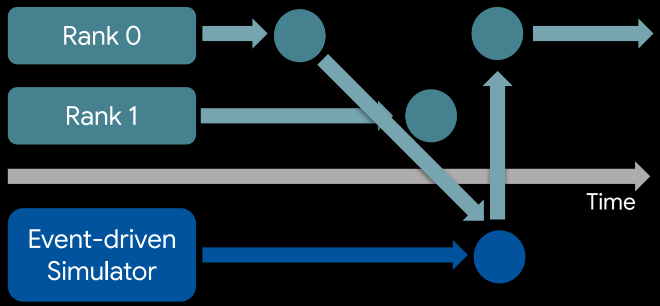
Solution: Rollback

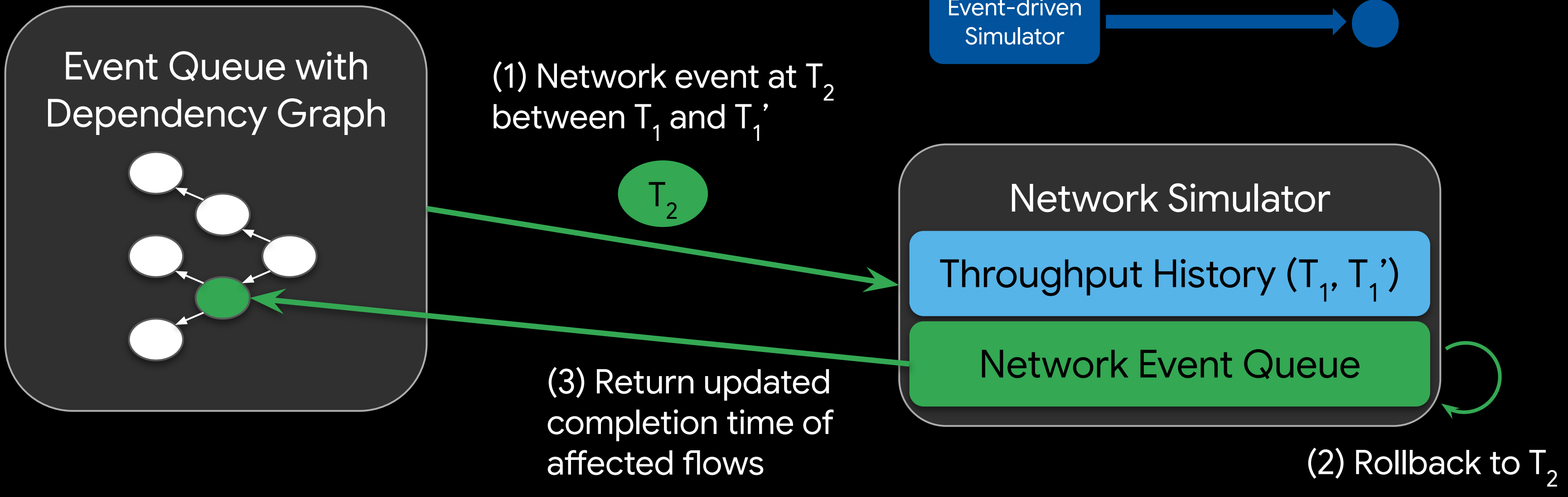


Solution: Rollback

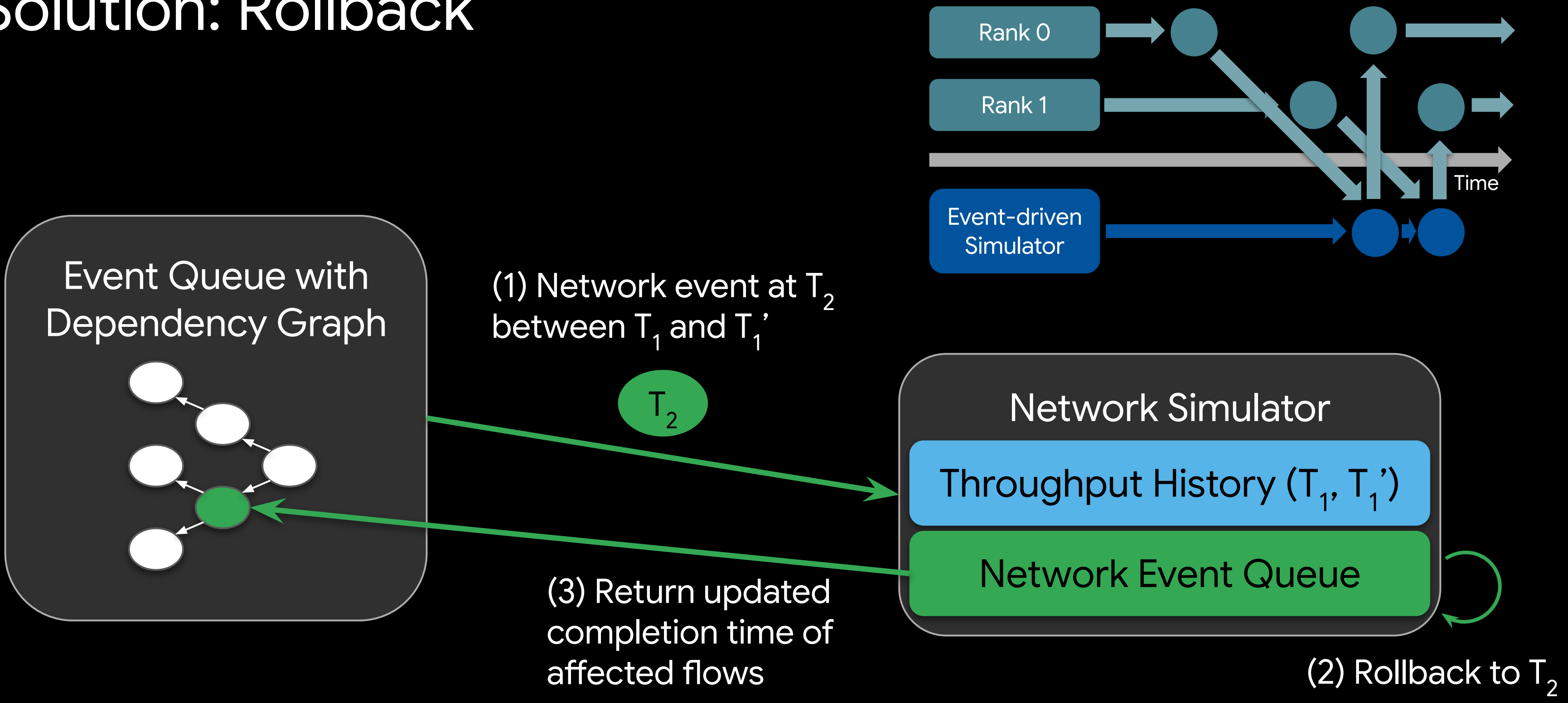


Solution: Rollback

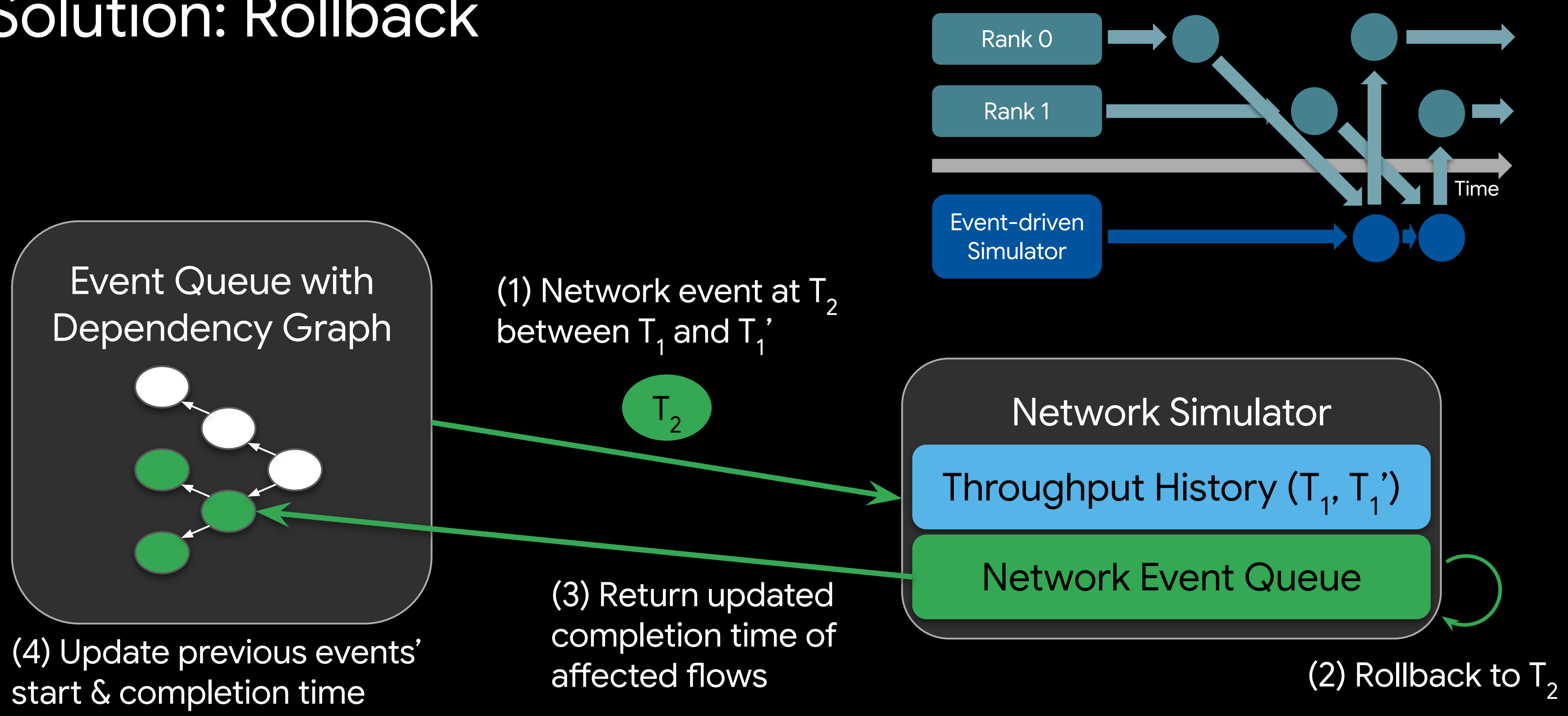




Solution: Rollback



Solution: Rollback



Emulation of NCCL & CUDA Runtime behaviors

GPU Memory Tracking

Track cudaMalloc & cudaFree, returning errors when exceeding configured capacity.

CUDA Streams & Events

Precise management of dependencies and synchronizations, including overlaps.

NCCL Communications

Handling complex synchronizations required for NCCL setup and collective operations.

Small User Effort: Unmodified frameworks

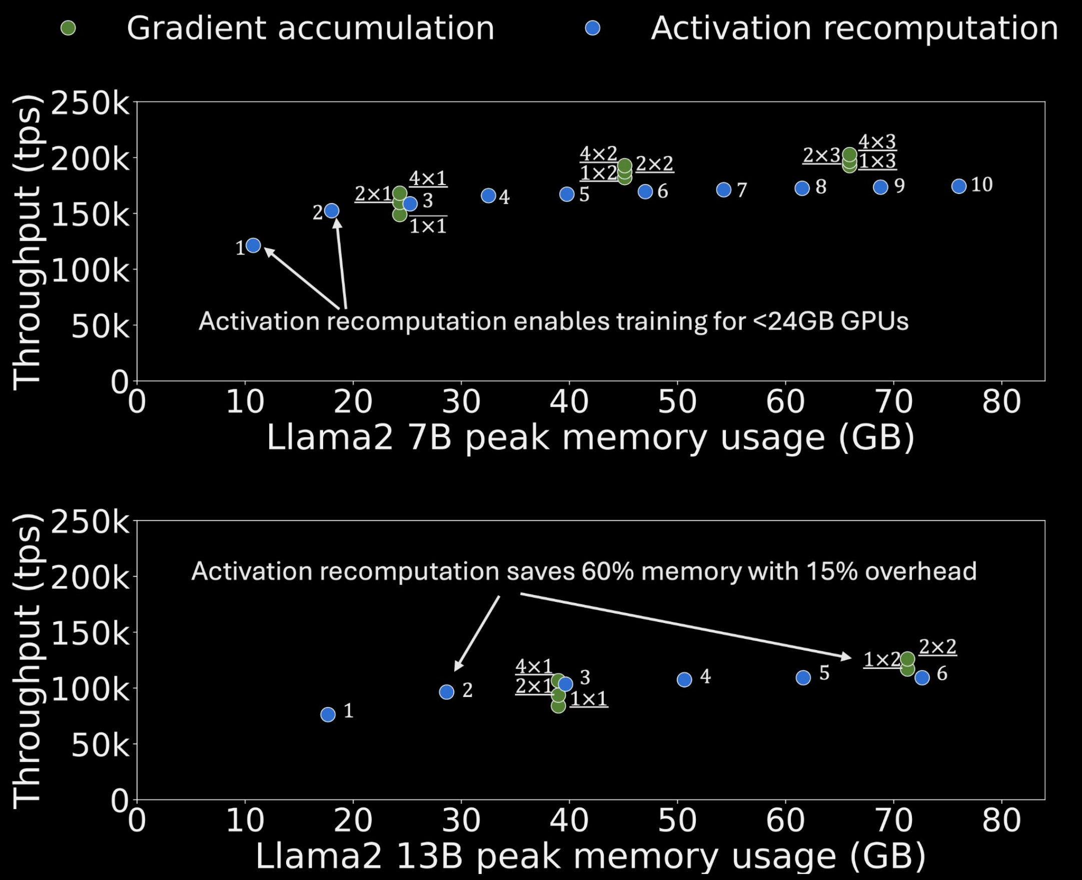
```
## install-requirements.sh
pip install megatron-core deepspeed torchtitan

## train.py
+from phantora_utils import (
+  enable_function_tracer,
+  disable_function_tracer,
+)

......

if __name__ == “__main__”:
+   enable_function_tracer()
    train()
+   disable_function_tracer()
```

Small User Effort: Logging as-is

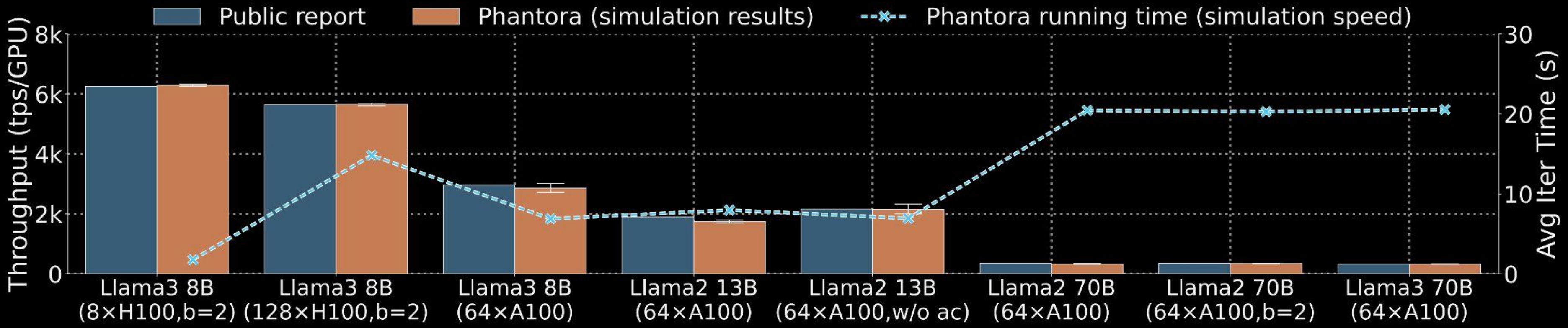
```python
logger.info(
    f"step: {train_state.step:2}  "
    f"loss: {global_avg_loss:7.4f}  "
    f"memory: {gpu_mem_stats.max_reserved_gib:5.2f}GiB"
    f"({gpu_mem_stats.max_reserved_pct:.2f}%)  "
    f"wps: {round(wps):,}  "
    f"mfu: {mfu:.2f}%{color.reset}"
)
```

```
[rank0]:2024-09-19 14:46:00,990 - root - INFO - step:  5  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,876  mfu: 53.38%
[rank0]:2024-09-19 14:46:37,206 - root - INFO - step: 10  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,887  mfu: 53.60%
[rank0]:2024-09-19 14:47:13,114 - root - INFO - step: 15  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,902  mfu: 53.87%
[rank0]:2024-09-19 14:47:47,622 - root - INFO - step: 20  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,904  mfu: 53.90%
[rank0]:2024-09-19 14:48:22,964 - root - INFO - step: 25  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,905  mfu: 53.92%
[rank0]:2024-09-19 14:48:58,821 - root - INFO - step: 30  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,904  mfu: 53.90%
[rank0]:2024-09-19 14:49:34,490 - root - INFO - step: 35  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,886  mfu: 53.57%
[rank0]:2024-09-19 14:50:30,248 - root - INFO - step: 40  loss:  0.0000
memory: 41.56GiB(51.95%)  wps: 2,894  mfu: 53.72%
```

Case Study: Tuning Megatron config under limited VRAM

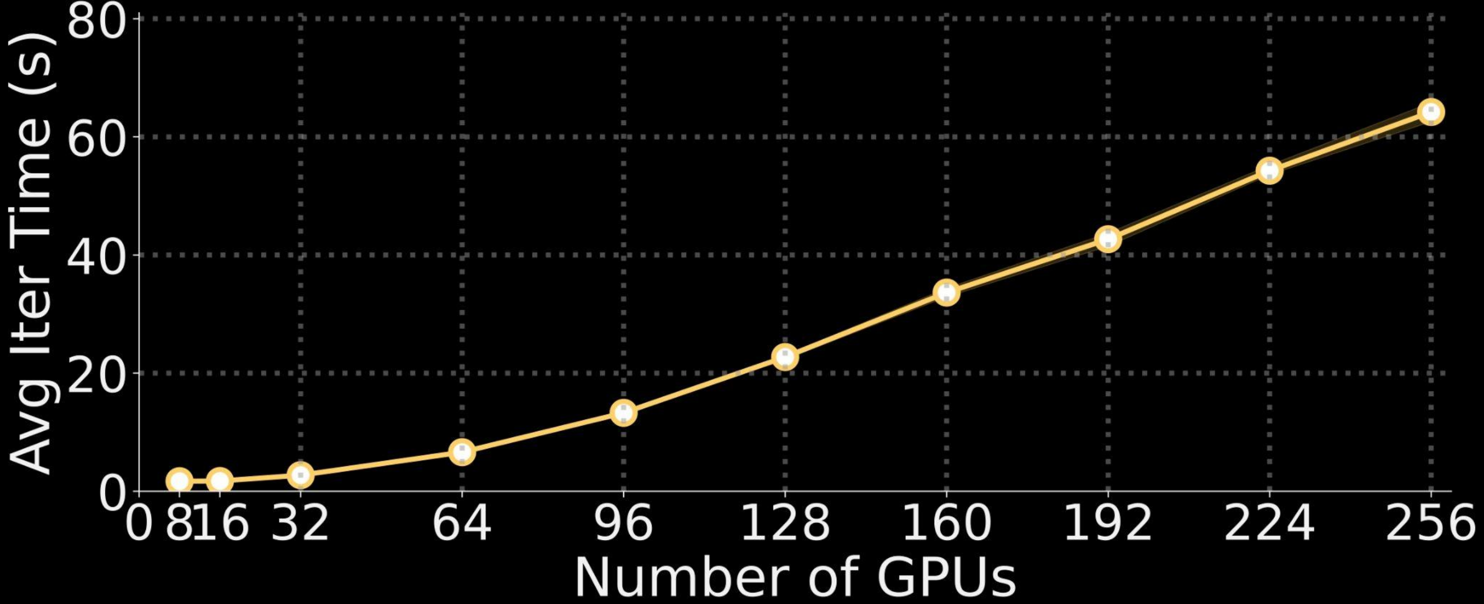


Evaluation: Accuracy and Speed



Phantora achieves 2.9% average error and 8.5% maximum error simulating TorchTitan's benchmark with 1×H200 or 1×A100

Evaluation: Scalability



Speed of Phantora (Megatron, Llama2 7B, simulated with 1×H200)

Discussions

- Tuning existing systems and exploring new system ideas (parallelizations, fault tolerance mechanism, hardware co-design, ...)
- Further improve network simulator: Other methods should be able to apply as Phantora still uses standard discrete event simulation.
- Value-dependent performance is challenging to incorporate, especially RL training performance, which heavily depends on response lengths.

Conclusion

- Current ML system performance simulation methods requires framework reimplementation. Direct reuse of framework code is achievable via hybrid simulation.

- Phantora provides high-fidelity estimation with small user effort and almost complete support of framework features.

- Phantora is publicly available at https://github.com/QDelta/Phantora.