NanoFlow: Towards Optimal Large Language Model Serving Throughput

Kan Zhu, *University of Washington;* Yufei Gao, *Tsinghua University and University of Washington;* Yilong Zhao, *University of Washington and University of California, Berkeley;* Liangyu Zhao, *University of Washington;* Gefei Zuo, *University of Michigan;* Yile Gu and Dedong Xie, *University of Washington;* Tian Tang and Qinyu Xu, *Tsinghua University and University of Washington;* Zihao Ye, Keisuke Kamahori, and Chien-Yu Lin, *University of Washington;* Ziren Wang, *Tsinghua University and University of Washington;* Stephanie Wang, Arvind Krishnamurthy, and Baris Kasikci, *University of Washington*

Yiqiao Lin

UCDAVIS COMPUTER SCIENCE

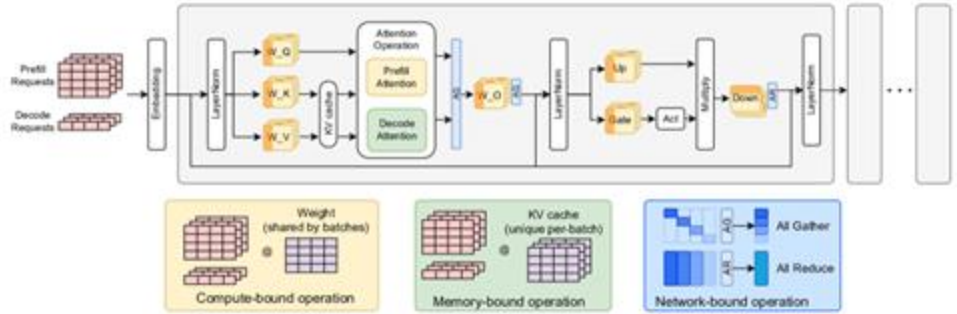
Background

Background -LLM Serving at Scale

- LLM usage has exploded (e.g., ChatGPT's hundreds of millions of users) on thousands of GPUs
- Throughput (tokens/sec) is critical for cost and performance
- LLMs are huge (billions of parameters) and use self-attention with large KV-caches
- Conventional wisdom: inference is memory-bound (large models & attention cache).
- Common solutions use inter-device parallelism (data, tensor, pipeline), but these do not overlap resources within a GPU.

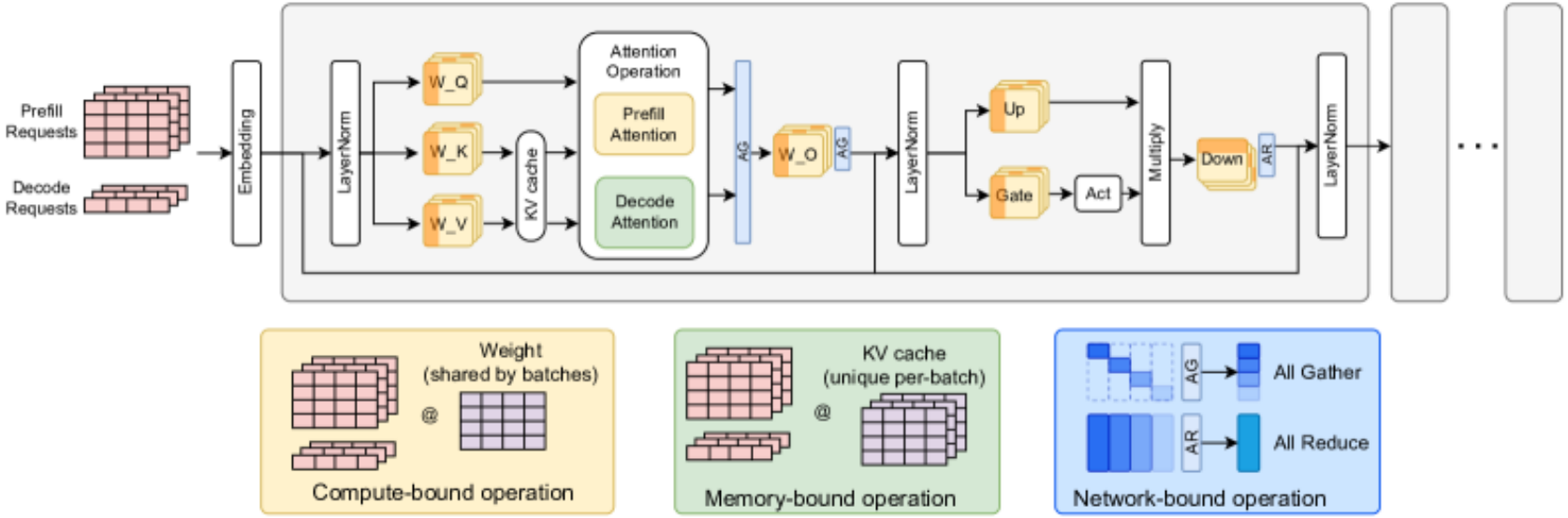
Background - Resource Bottlenecks in LLM Inference

- Inference ops vary: GEMMs (compute-bound) vs attention (memory/network-bound) vs all-reduce (network)
- For a single batch, each op saturates its bottleneck resource (~80% individually), but sequential execution yields only ~40% total GPU compute utilization
- They find modern LLM serving compute-bound overall (GEMMs dominate)
- Sequential execution means while memory/network ops run, expensive compute units sit idle.



UCDAVIS COMPUTER SCIENCE

Background - Resource Bottlenecks in LLM Inference



UCDAVIS COMPUTER SCIENCE

Background -Limits of Current Serving Engines

- Common inter-device parallelism schemes include data parallelism, tensor (model) parallelism, and pipeline parallelism. But the GPU operation still largely sequential.
- none has achieved concurrency at the operation level inside the GPU.
- Goal: keep the GPU's compute units as busy as possible, by finding opportunities to overlap other operations.

NanoFlow

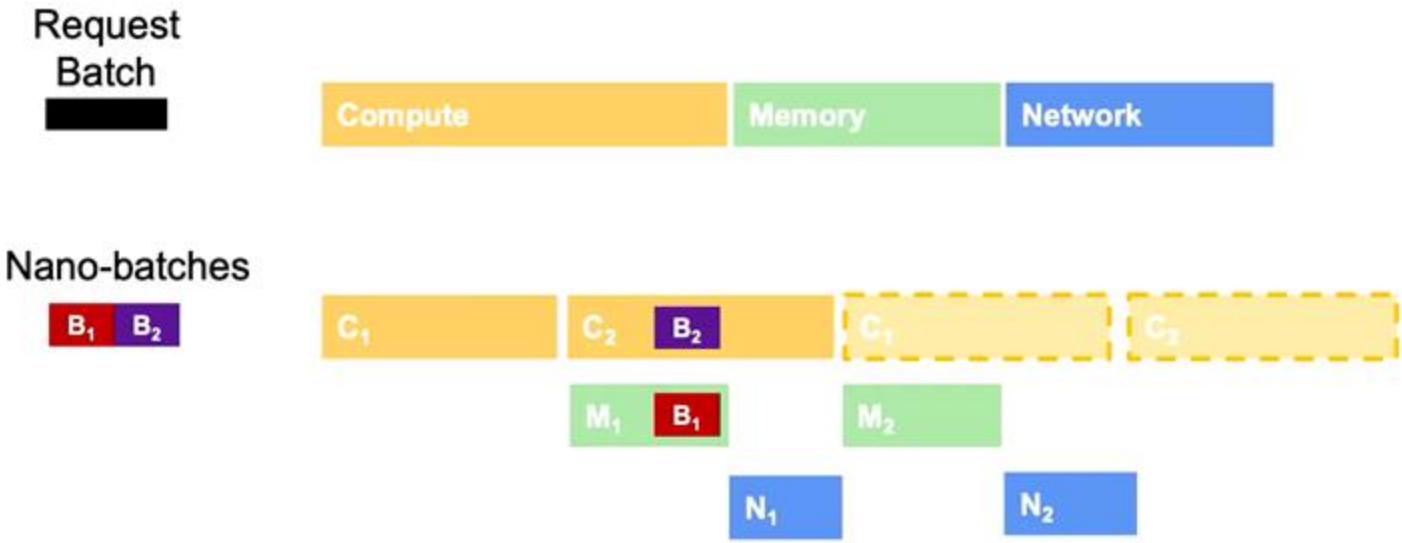
NanoFlow - Key Idea

- NanoFlow's core insight: overlap heterogeneous ops within a GPU by running them concurrently
- Split each input batch into nano-batches at the granularity of individual operations
- Duplicate operations for each nano-batch so that previously sequential ops can run in parallel.
- E.g., split a batch of requests so compute-heavy and memory-heavy operations operate on different parts simultaneously.

NanoFlow - Nano-Batching & Execution Units

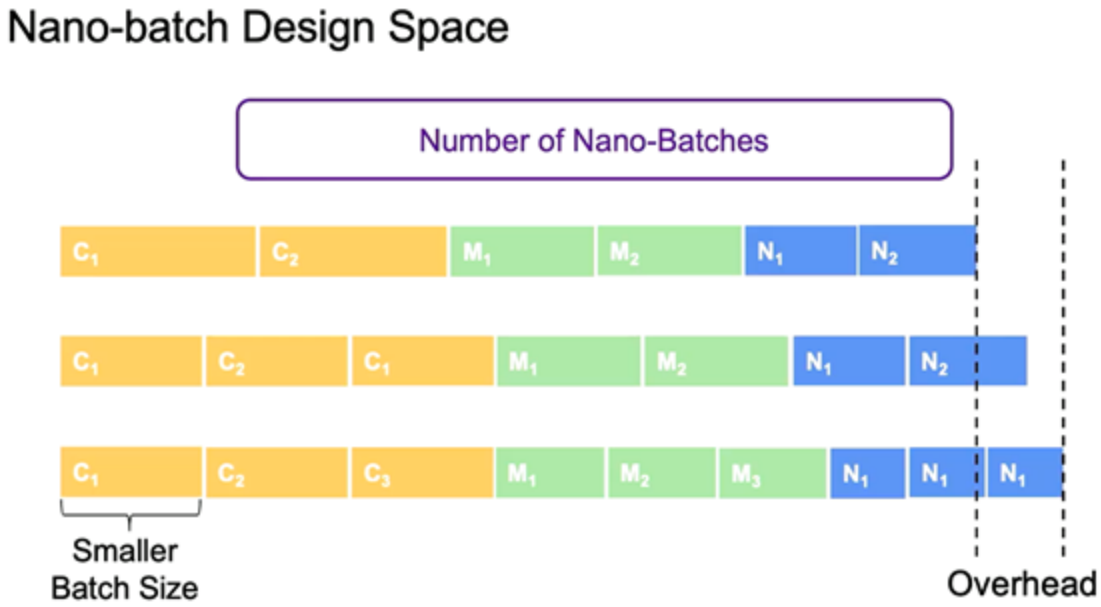
- Nano-batch mechanism: Each large batch is divided into smaller “nano-operations”, each handling a subset of data.
- Execution-unit scheduling: GPU SMs are partitioned so different operations run on different SM subsets
- Example: Assign most SMs to the compute-intensive GEMM nano-ops, leaving some SMs for memory-bound attention ops in parallel.
- Result: The GPU’s compute units are never idle during memory/network ops, boosting utilization

NanoFlow - Nano-Batching & Execution Units



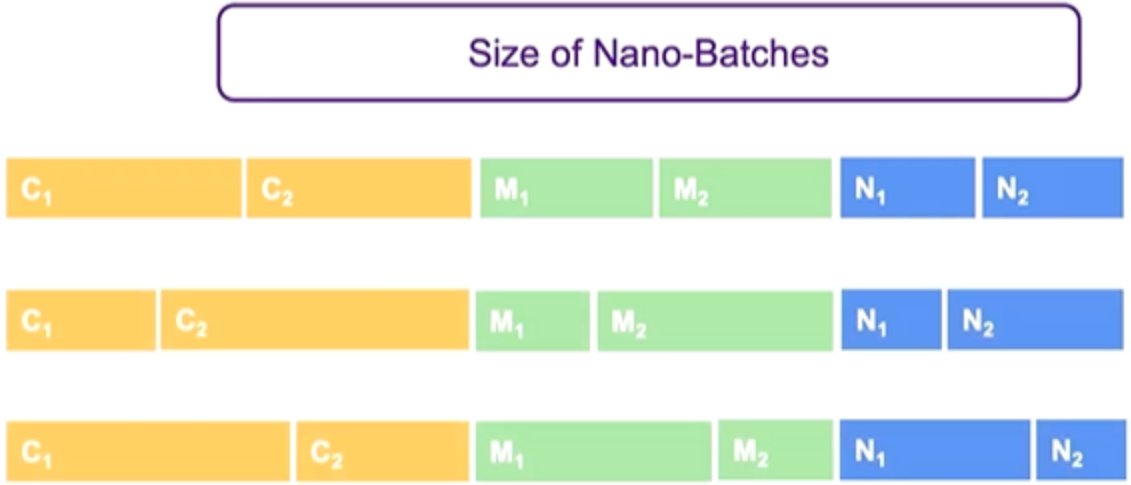
UCDAVIS
COMPUTER SCIENCE

NanoFlow - Nano-Batching & Execution Units



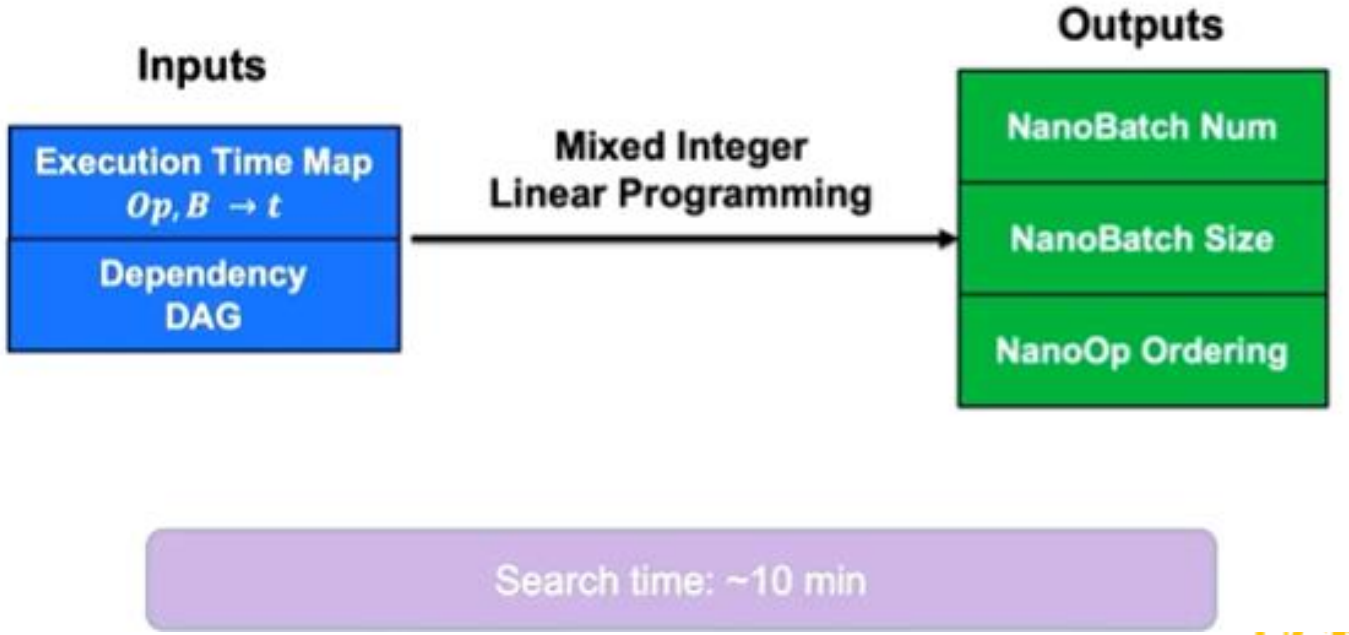
UCDAVIS
COMPUTER SCIENCE

NanoFlow - Nano-Batching & Execution Units



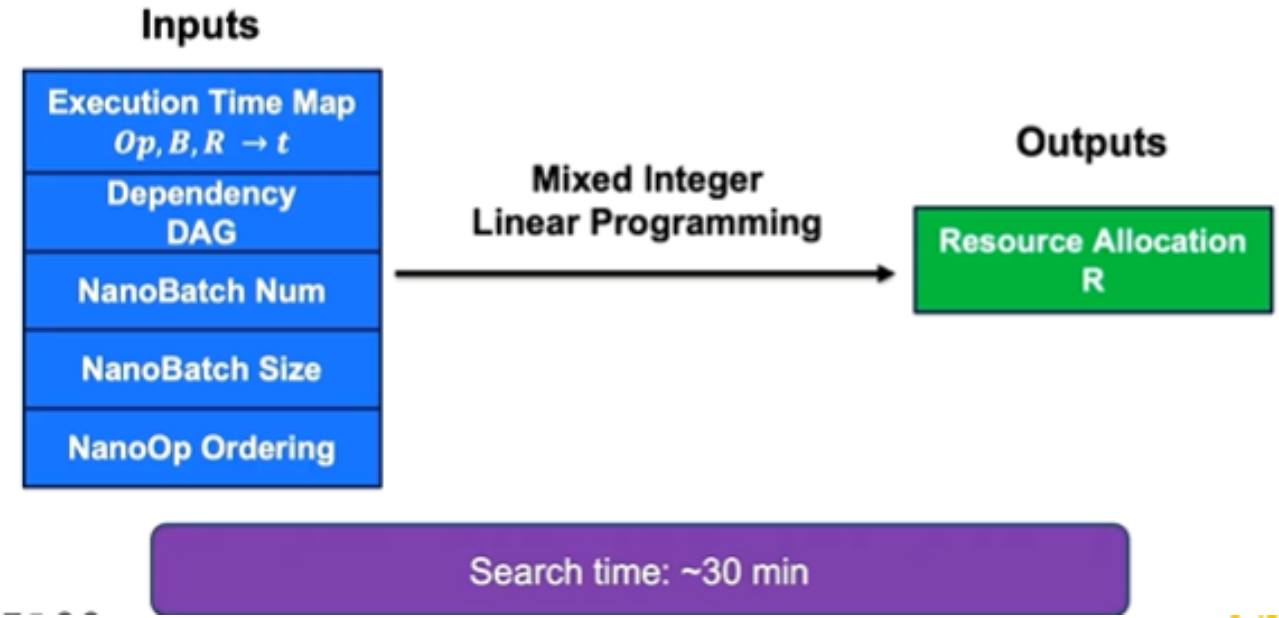
UCDAVIS
COMPUTER SCIENCE

NanoFlow - intra-device pipeline scheduler



UC DAVIS
COMPUTER SCIENCE

NanoFlow - intra-device pipeline scheduler



NanoFlow - System Architecture

- Batch Scheduler: uses an asynchronous scheduling loop to reduce CPU overhead.
- KV-Cache Manager: Offloads per-request key-value cache to CPU/SSD for long dialogues
- The intra-device pipeline scheduler: find optimal nano-batch sizes and SM allocations

UCDAVIS
COMPUTER SCIENCE

Evaluation

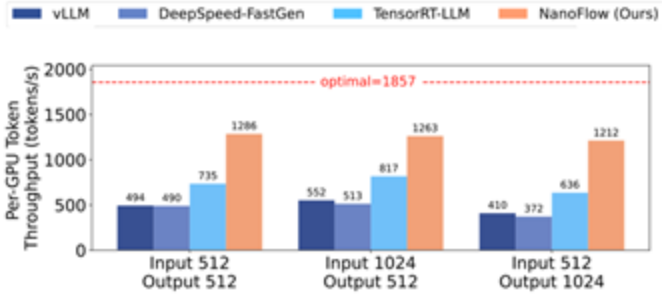
Evaluation - Experimental Setup

- a single node with 8 NVIDIA A100 GPUs (DGX).
- LLaMA-2-70B (70 billion parameters), LLaMA-3-70B, Mixtral-87B, LLaMA-3-8B, QWen2-72B, and Deepseek-67B.
- Workload:ShareGPT, LMSys Chat, and Splitwise datasets.
- Baseline: vLLM , DeepSpeed-FastGen, and TensorRT-LLM.

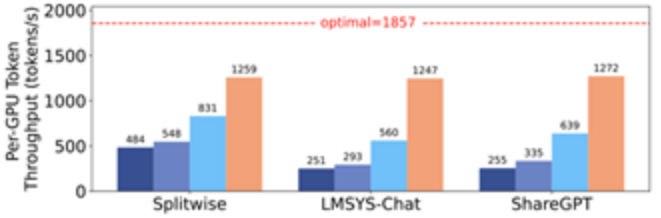
UCDAVIS
COMPUTER SCIENCE

Evaluation - Throughput

- NanoFlow ~1.9× throughput of vLLM/DeepSpeed-FastGen/TensorRT baselines
- Reaches up to 68.5% of the theoretical optimal token/s rate
- Applies to both synthetic constant-length and real workload traces.

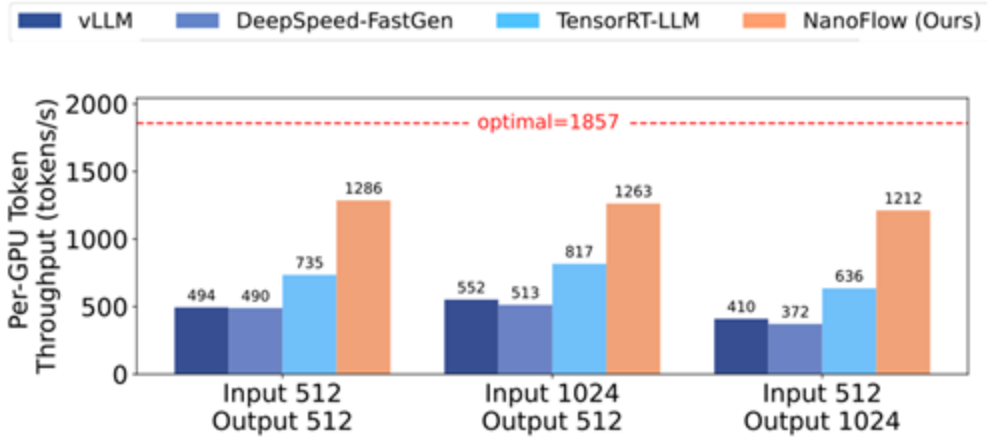


(a) LLaMA-2-70B, 8 GPU, TP=8, Constant Input & Output Length

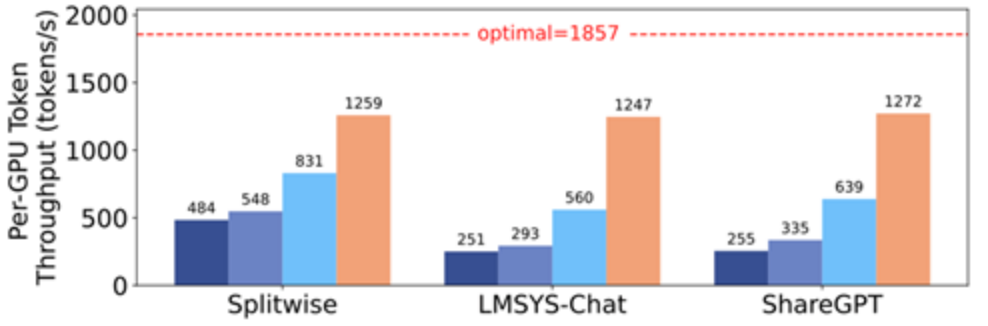


(b) LLaMA-2-70B, 8 GPU, TP=8, Input & Output Length from Dataset

UCDAVIS COMPUTER SCIENCE



(a) LLaMA-2-70B, 8 GPU, TP=8, Constant Input & Output Length

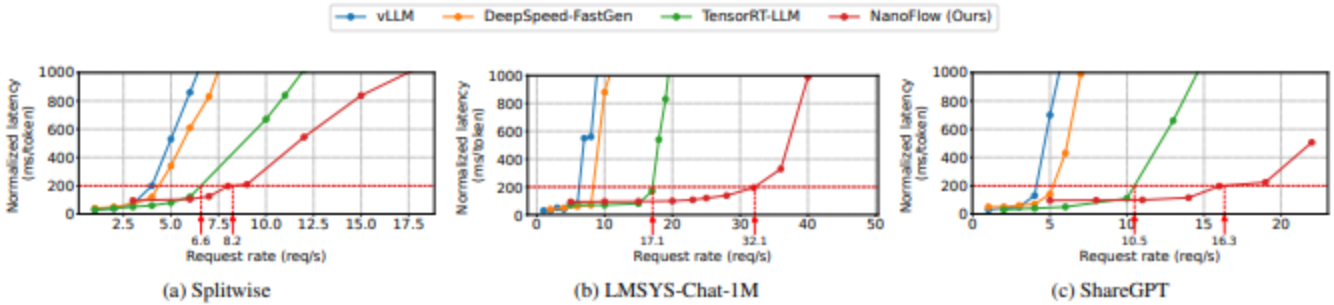


(b) LLaMA-2-70B, 8 GPU, TP=8, Input & Output Length from Dataset

UCDAVIS COMPUTER SCIENCE

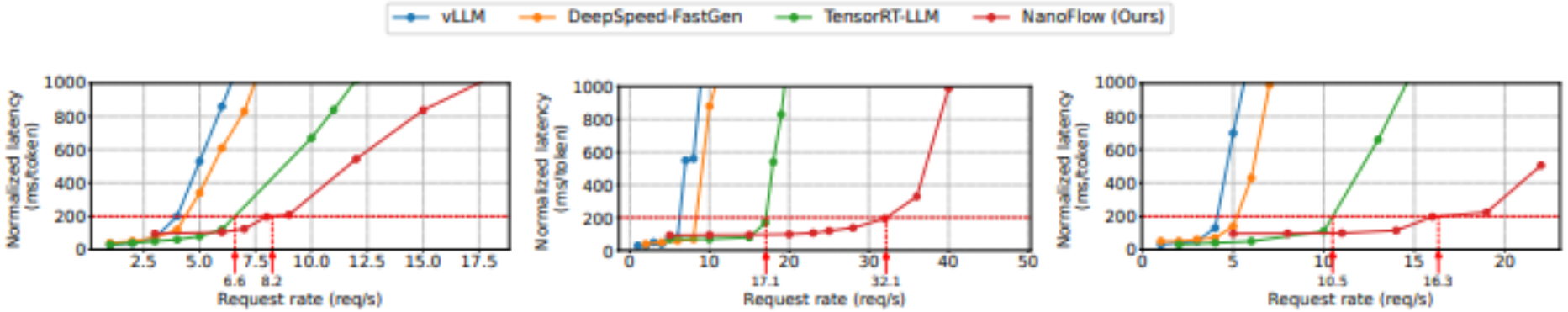
Evaluation – Latency

- Result: NanoFlow sustains a much higher request rate before saturation, at similar low latency.
- At low rates, NanoFlow's latency ≈ best baseline; at high rates, it maintains service (up to 1.64× higher QPS)
- Demonstrates that NanoFlow's gains do not come at cost of much higher



(a) Splitwise (b) LMSYS-Chat-1M (c) ShareGPT

Evaluation – Latency



(a) Splitwise (b) LMSYS-Chat-1M (c) ShareGPT

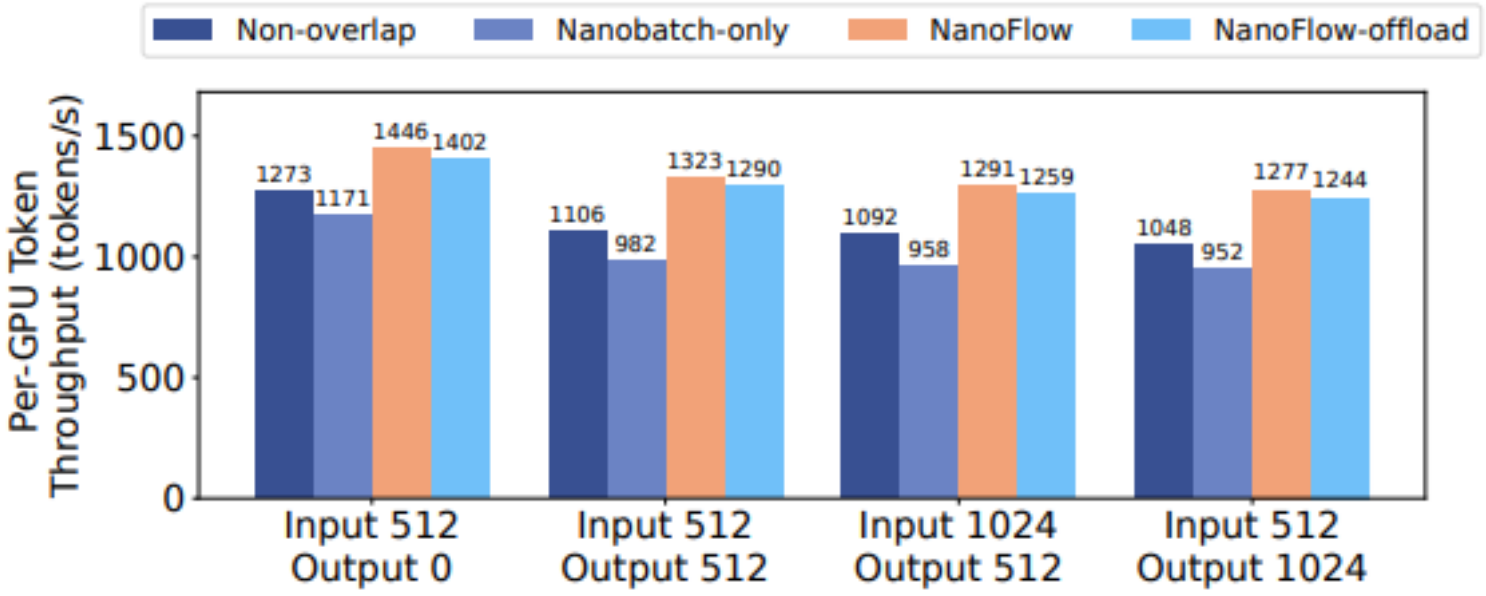
UCDAVIS COMPUTER SCIENCE

Evaluation – Ablation Study

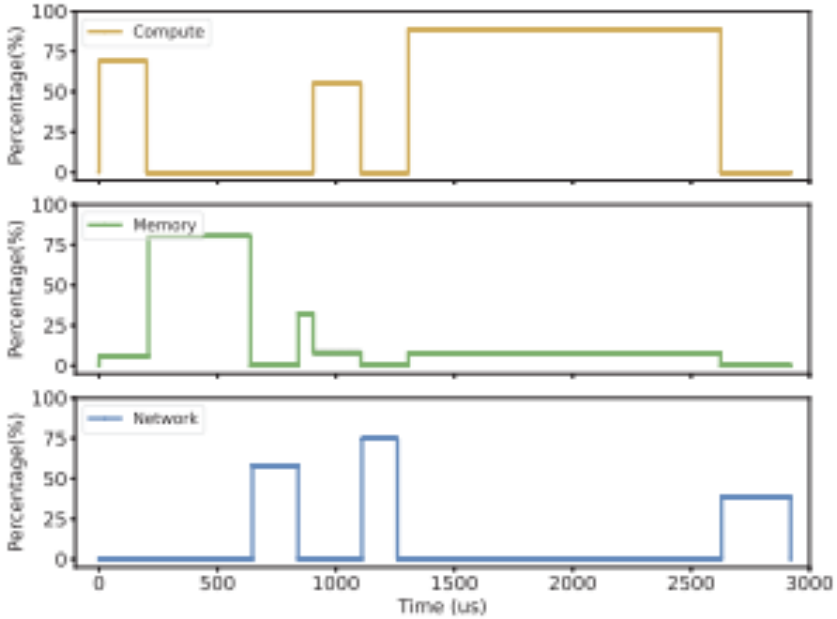
- Configurations compared: Non-overlap baseline (no splitting), Nano-batch only (split but no overlap), and NanoFlow (full overlap).
- Nano-batch overhead: Splitting alone (without overlap) incurs some overhead (slower than non-overlap) due to fragmentation
- Overlap benefit: Enabling overlapped scheduling yields large speedups over baseline.
- Offloading: Adding KV offload slowed pipeline by a small %, but greatly improves multi-turn throughput.

UCDAVIS
COMPUTER SCIENCE

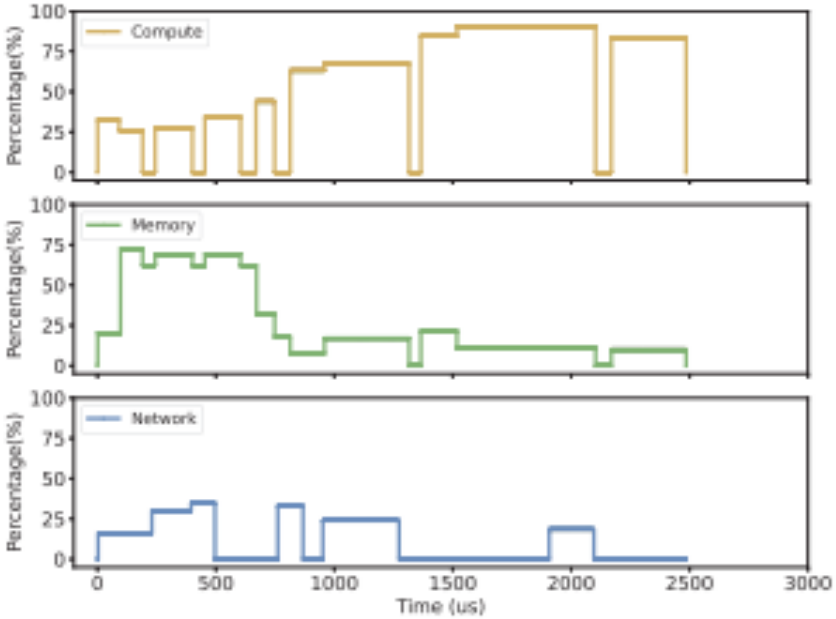
Evaluation – Ablation Study



Evaluation – Resource Utilization

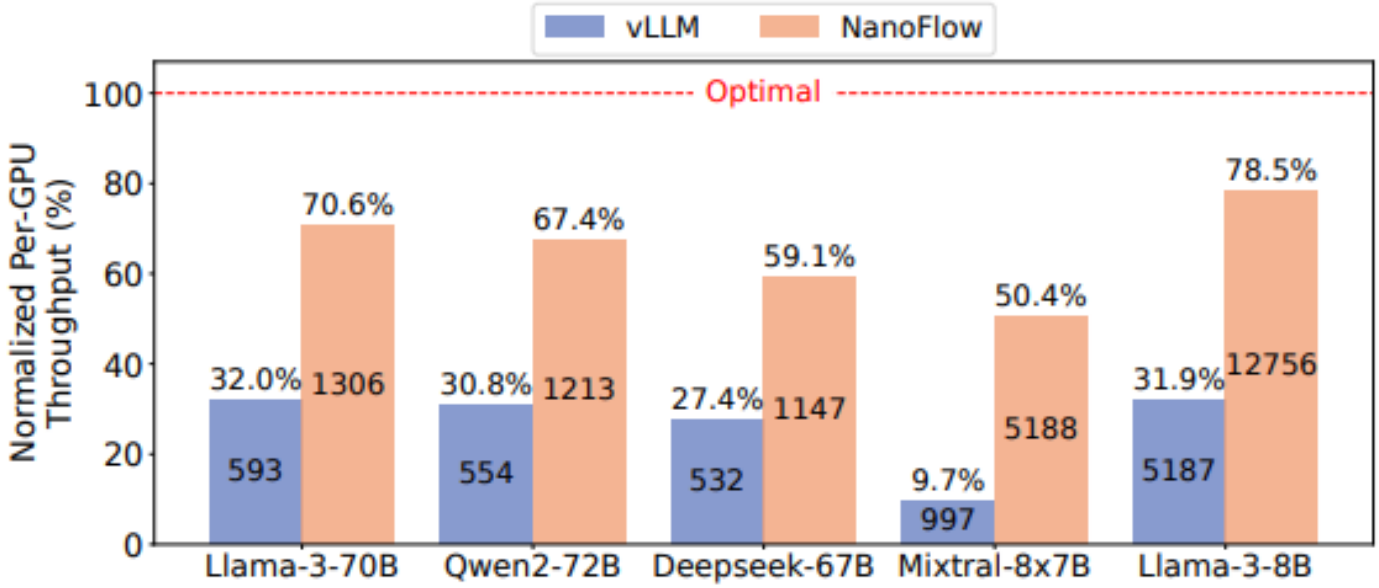


(a) Non-overlap pipeline resource usage



(b) NanoFlow resource usage

Evaluation – Portability to Other Models



Conclusion

Conclusion

- NanoFlow introduces operation-level pipelining in LLM serving: overlapping compute, memory, and network within each GPU
- Through auto-tuned nano-batches and execution-unit scheduling, it achieves near-optimal throughput across models
- Key takeaway: Intra-device parallelism is a new frontier for high-throughput LLM inference.

UCDAVIS
COMPUTER SCIENCE

Thank you

UCDAVIS
COMPUTER SCIENCE