Dynamic Space-Time Scheduling for GPU Inference

Paras Jain, Simon Mo, Ajay Jain§, Harikaran Subbaraj, Rehan Durrani
Alexey Tumanov, Joseph Gonzalez, Ion Stoica

riselab UC Berkeley §MIT

Contact Paras Jain
paras_jain@berkeley.edu

SCAN ME

Background/Context:
- ML inference increasingly important for soft real-time latency-sensitive applications

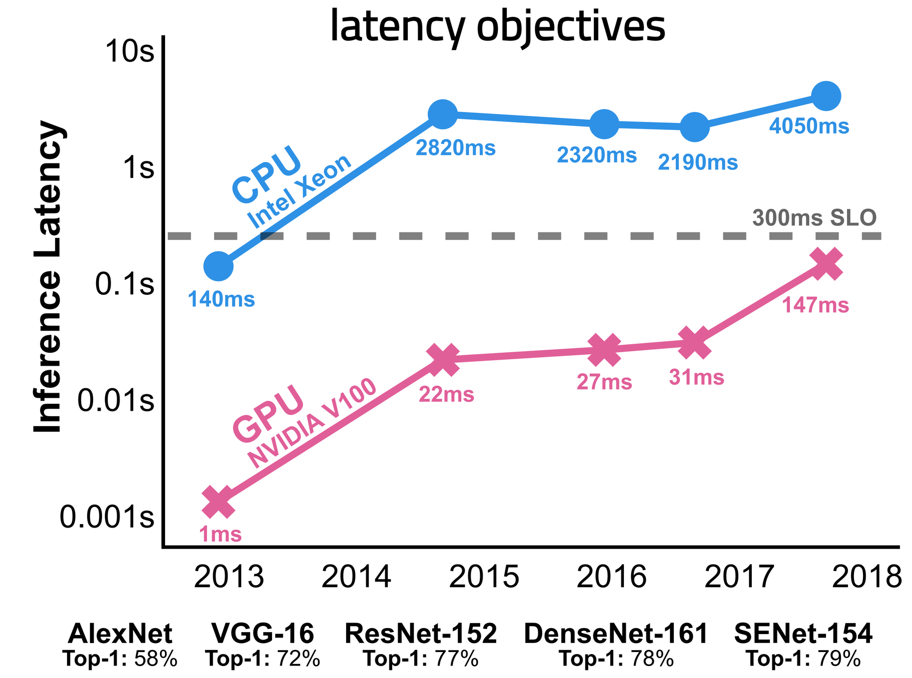
Problem:
- Model complexity increases —> latency increases; too slow for soft-real time applications
- GPU —> more attractive for inference, suffers low utilization (under latency constraints)

Solution:
- *State of the art*: temporal multiplexing
- *Proposal*: multiplex GPUs across multiple models and time

Motivation: Online inference leads to low GPUs utilization

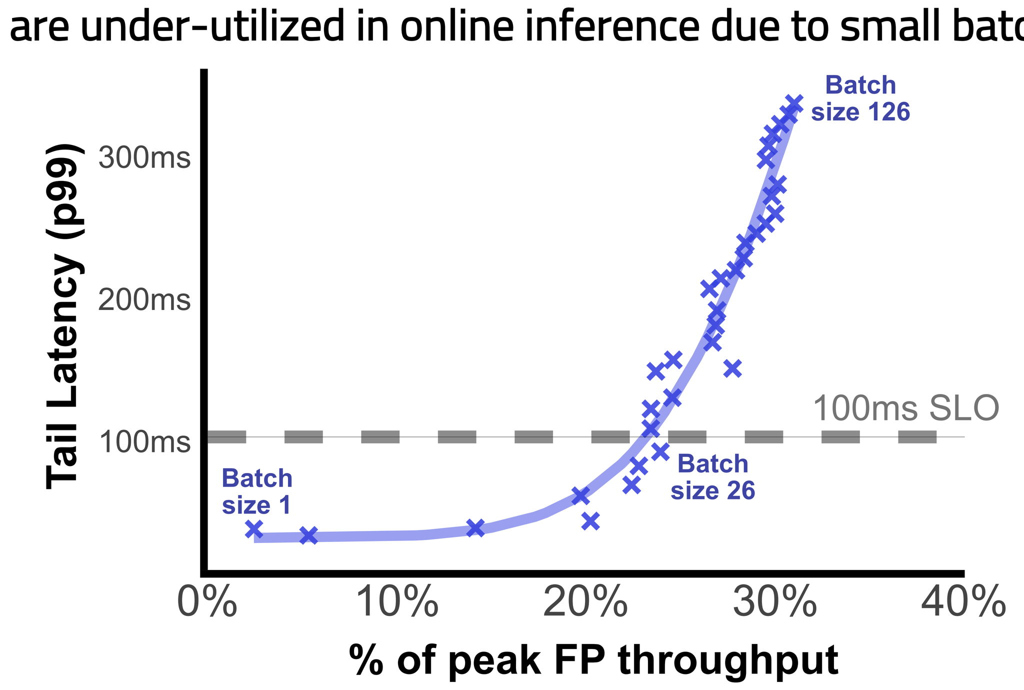
- Due to growing DNN model complexity, **CPUs cannot meet** online inference latency requirements

As models continue to grow, GPUs are the only way to meet online inference latency objectives



- Small batches common in inference leads to **poor GPU utilization** and **low resource-efficiency**

GPUs are under-utilized in online inference due to small batch-sizes



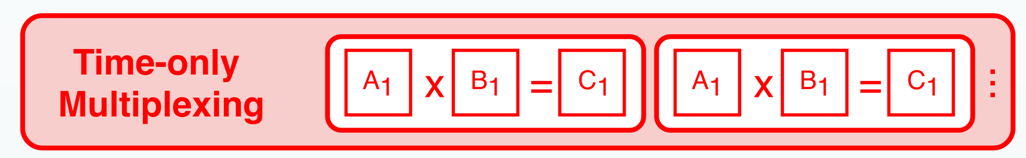
- We explore **spatial and temporal multiplexing** techniques to improve **GPU utilization**

Key requirements for GPU kernel multiplexing

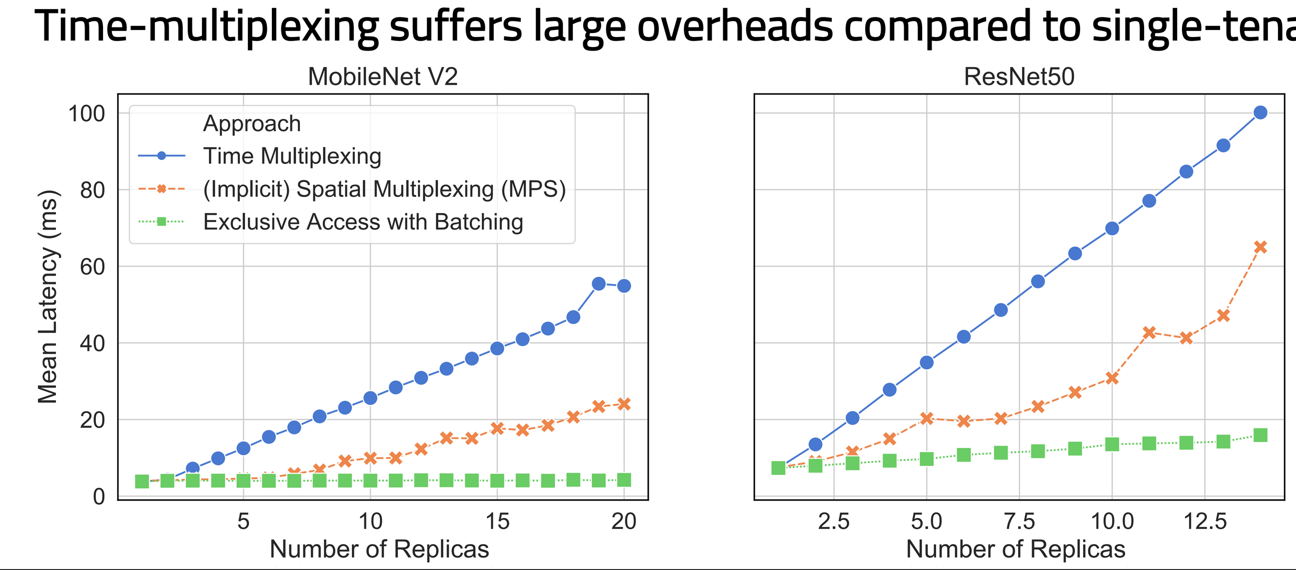
- Runtime performance must be **predictable**
- Multiplexing should increase **resource-efficiency**
- Models on a single GPU should have **inter-tenant isolation**

Time-only multiplexing: poor resource-efficiency

- **Time-multiplexing**: on device scheduler enables interleaved execution of multiple CUDA contexts (no parallel execution)
- **Pro:** Guaranteed isolation between tenants and predictability
- **Con:** Sharply degraded throughput and increased latencies

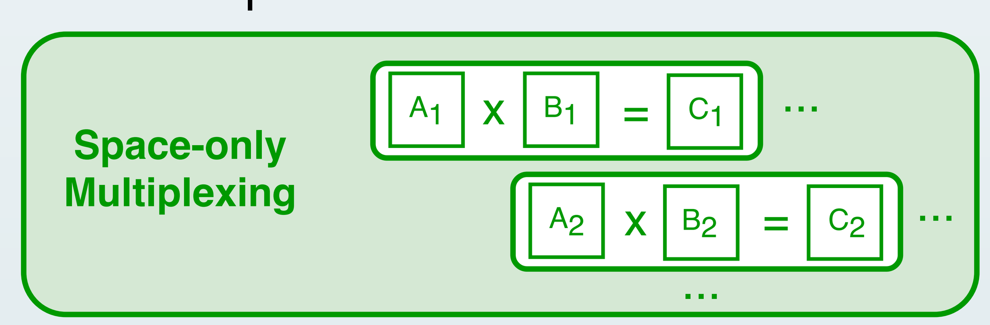


Time-multiplexing suffers large overheads compared to single-tenancy

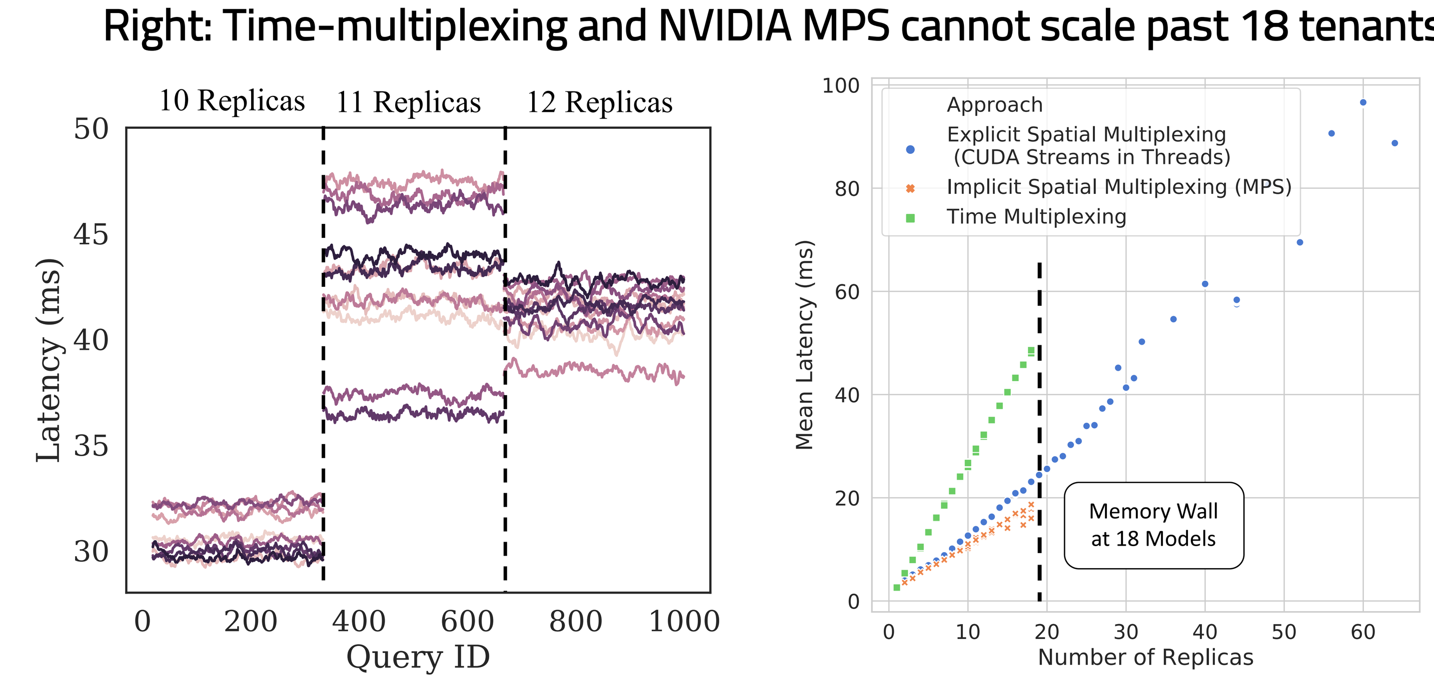


Space-only multiplexing: poor predictability and isolation

- Spatial-multiplexing: NVIDIA Multi-Process Service to partition kernel launches across multiple CUDA Streams
- **Pro:** Kernels can execute in parallel
- **Con:** Unpredictable performance and lack of isolation

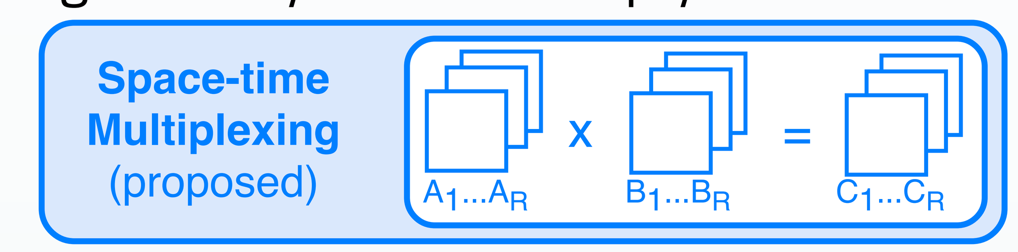


Left: As we add replicas, space multiplexing latencies are unpredictable
Right: Time-multiplexing and NVIDIA MPS cannot scale past 18 tenants

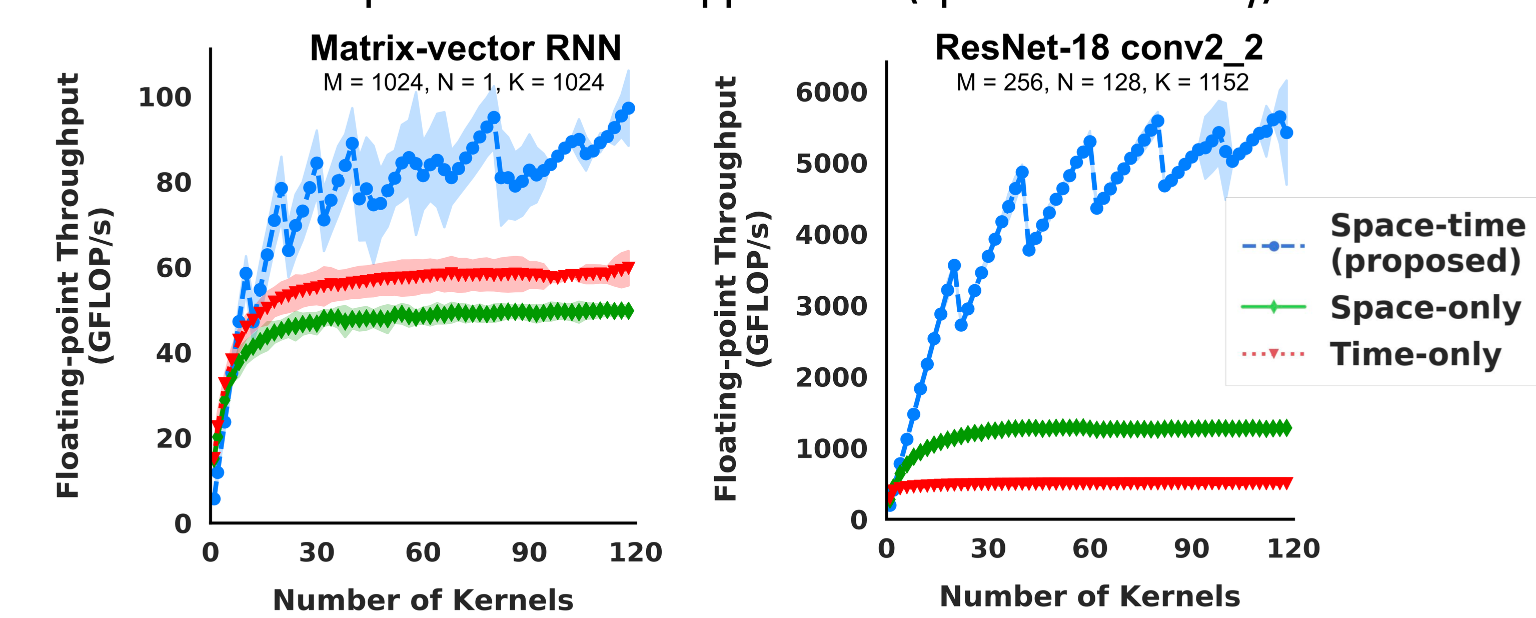


Proposed solution: Space-and-Time multiplexing

- **Space-Time scheduling:** Inter-model kernel batching via a software scheduler
- Preliminary benchmark of approach: batch multiple GEMMs
- `cublasSgemmBatched` provides high-throughput multiplexing of many matrix multiply kernels

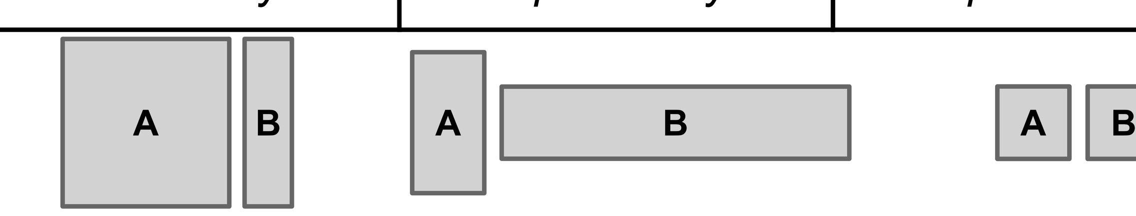


Space-time multiplexing achieves higher FP32 throughput compared to current approaches (space or time only).



Space-Time has 2.5-4.6x throughput speedups over prior art

Number of Kernels	Matrix-vector RNN, 1024 hidden units M = 512, N = 1, K = 512	ResNet-18 conv2_2 M = 256, N = 128, K = 1152	Square SGEMM M = N = K = 256
10	1.21x	1.68x	2.42x
20	2.14x	2.88x	3.47x
2 - 120 (geomean)	**2.48x**	**3.23x**	**4.63x**
Speedup over?	*Time-only*	*Space-only*	*Space-only*



Conclusions: A large opportunity gap in performance

- There still exists a **5x performance gap** in utilization
- We show that space-time multiplexing can drive up to **2.5x-4.6x** throughput speedups
- Utilization issues are **much worse for NLP** (TPU gets 3% util.)
- We believe inter-model kernel scheduling will be a key technique to leverage fast-growing DNN accelerators
- Our preliminary results show promise for a dynamic space-time scheduler for efficiency, predictability and isolation.