Transparent GPU Sharing in Container Clouds for Deep Learning Workloads

Bingyang Wu and Zili Zhang, *Peking University;* Zhihao Bai, *Johns Hopkins University;* Xuanzhe Liu and Xin Jin, *Peking University*

Motivation

- Static binding of GPUs to DLT containers produces low utilization
- Existing SOTA solutions have tradeoffs that limit their usefulness
  - Application Layer Solutions - not generic / transparent
  - OS-layer solutions - inadequate utilization or introduce new issues
- Can a GPU sharing mechanism for DLT that be developed that is achieves:
  - Has high utilization
  - Is Transparent to applications
  - Doesn’t introduce other undesired after effects

	AntMan [6]	Salus [12]	PipeSwitch [13]	MPS [9]	MIG [14]	TGS
Transparency				✓	✓	✓
High GPU utilization	✓	✓				✓
Performance isolation	✓	✓	✓	✓	✓	✓
Fault isolation	✓		✓		✓	✓

Key insight 1: Sharing Compute

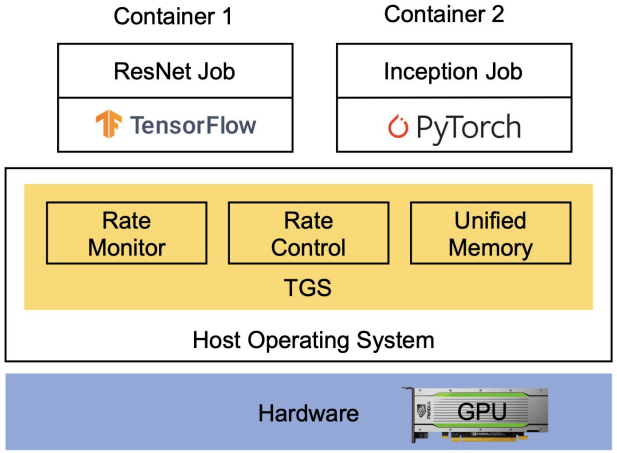
- The CUDA kernel arrival rate can be used as the heuristic for the performance of a job without application knowledge

- This heuristic can be used as a feedback signal to determine if high priority (performance) jobs are being impacted by low-priority (opportunistic) jobs that share the GPU with it.

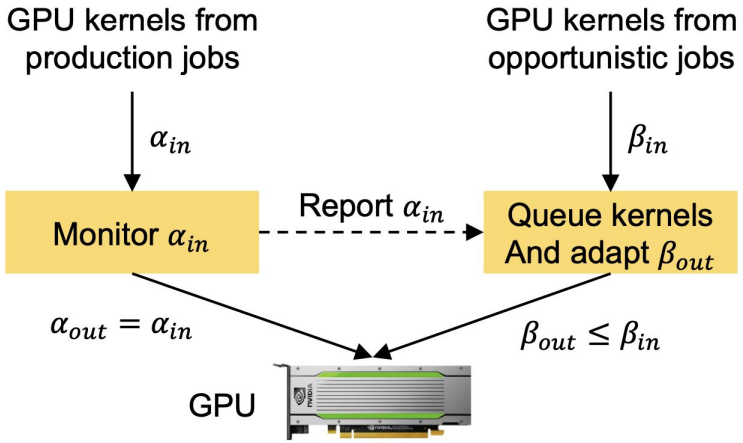
Key Insight 2 : Sharing Memory

- CUDA unified memory allows for virtualization of GPU memory
- By intercepting GPU memory allocations, GPU memory sharing can be transparent to the user.
- Enables performance isolation by paging in from / out to unified memory from GPU
- Additional benefit: supports memory oversubscription

System Details



TGS Architecture



Adaptive rate control

~3000 LOC C++ & Python integrated with Docker & Kubernetes

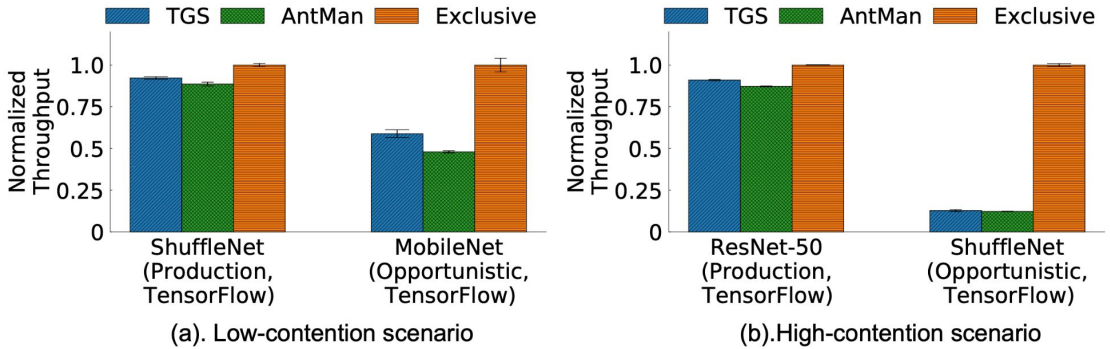
System Tradeoffs

- AIMD Adaptive Rate Control
  - Necessarily reduce performance of production job to find optimal sharing
  - Introduces new complexity in maintaining queues, monitoring and performing control computations
- Indirection with unified memory
  - Overhead of another layer of address translations and page fault expenses in GPU memory that did not exist beforehand.

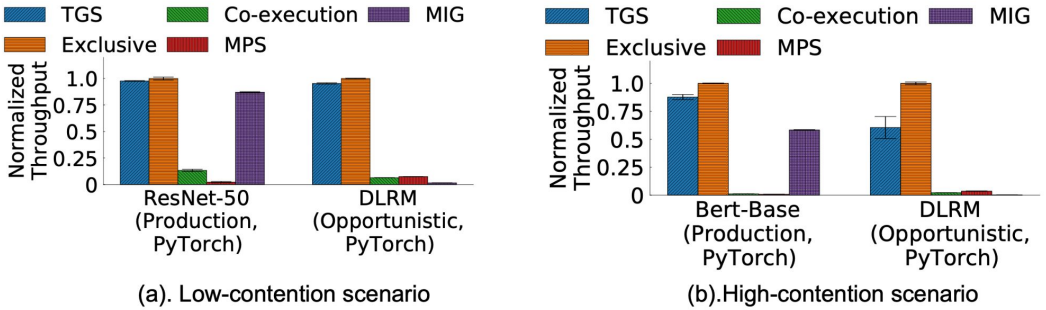
Evaluation Baselines

- AntMan: SOTA application layer solution (used by Alibaba)
- MPS: manually allocating memory
- MIG: manually set a partition configuration
- Exclusive: no sharing
- Co-execution: share GPU with no control

Evaluations: most significant improvement



TGS v.s. AntMan - transparency without performance degradation



Supports memory oversubscription with strong throughput

Evaluation: most disappointing result

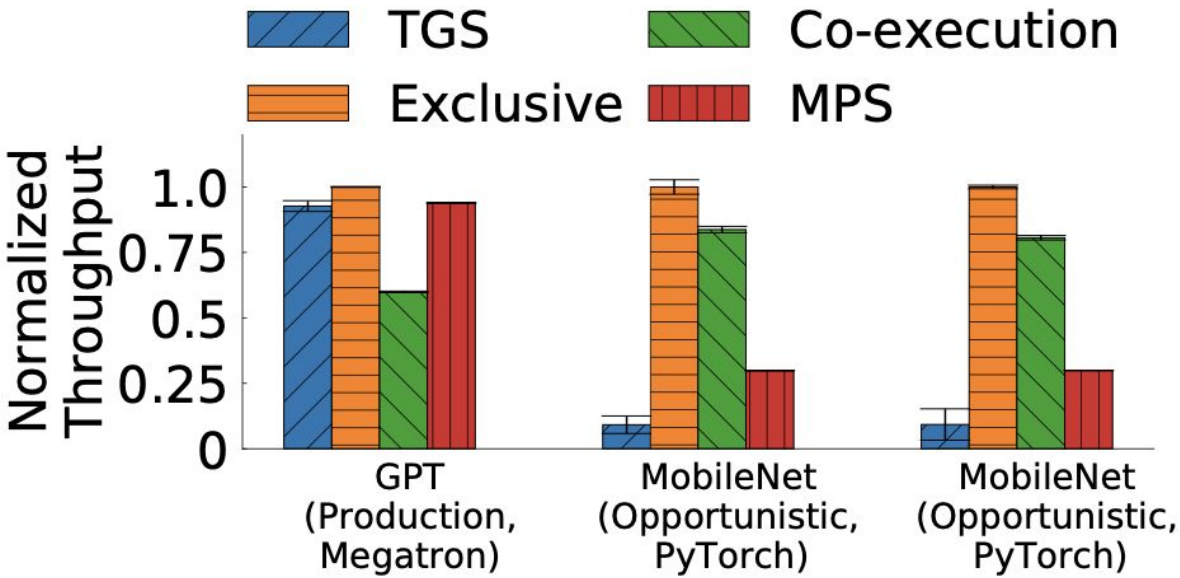


Figure 12: GPU sharing with the large model.

Discussion

Strengths

- Extremely rigorous evaluation
  - Tuning baselines to have the best performance possible
  - Testing different aspects in isolation AND using lots of different metrics e.g. JCT, throughput etc.
- Seems their system works extremely well
- Very strong motivation and outlining of their key differentiations (slide 3)
- Outlining a strawman and proceeding to their solution was both helpful and also demonstrated their process of thoughts (not typical in research)

Weaknesses

- Production v.s. Opportunistic is an ambiguous & coarse grained classification scheme - how realistic is this?
- Adaptive Rate Control description a bit hand-wavy
- How were batch sizes selected? Significant variation often without justification.
- Just NVIDIA? Is this an issue, at the very least needs to be more clear!
- Compute & Memory sharing distinction is not clear at all.