DAC Young Fellows

DESIGN AUTOMATION CONFERENCE
FROM CHIPS TO SYSTEMS — LEARN TODAY, CREATE TOMORROW

SET: Stream-Event-Triggered Scheduling for Efficient CUDA Graph Pipelines

Zhengxiong Li; ID: 690; Advisor: Tsung-Wei Huang, Umit Ogras; UW-Madison

Introduction

- Many workloads fail to achieve full GPU utilization, even with CUDA graph
- Idle intervals, called kernel gaps, often appear on GPU timeline between successive kernel executions

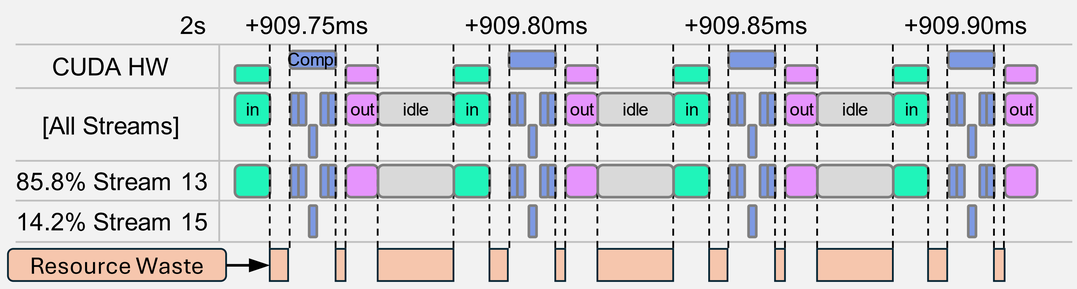

Fig. 1: Gaps between operations in Nsight profiler

Analysis

- Ideal case: $T_{ideal} = bt_{in} + t_k + t_{out}$
- Intra-batch overhead:
  - $t_{intra} = (b-1)t_{in-in} + t_{in-k} + \Delta t_k + t_{k-out}$
- Inter-batch overhead:
  - $t_{inter} = t_{batch2}^{start} - t_{batch1}^{end}$
- Wall clock:
  - $T_{measured} = T_{ideal} + t_{intra} + t_{inter}$

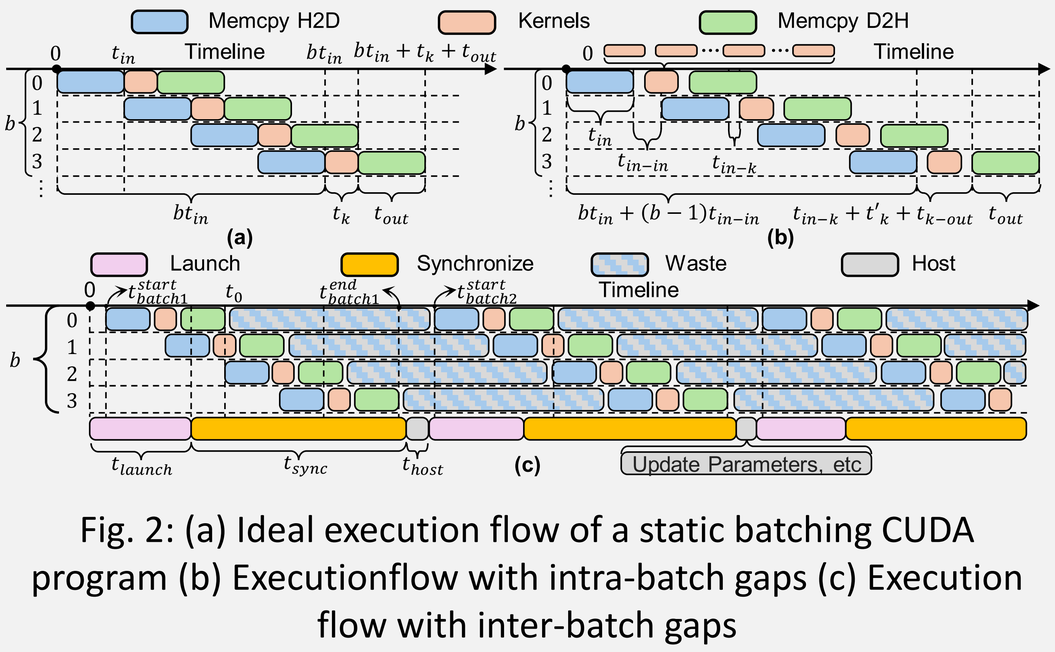

Fig. 2: (a) Ideal execution flow of a static batching CUDA program (b) Executionflow with intra-batch gaps (c) Execution flow with inter-batch gaps

SET

- Workers W_i, Per-worker job queue Q_i, Free worker pool W_{pool}, Submitter and dispatcher:

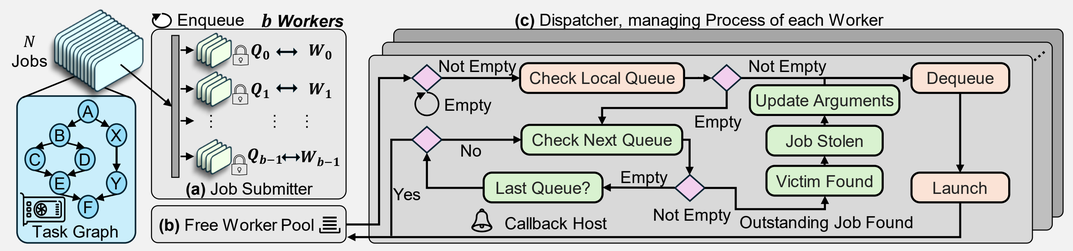

Fig. 3: Overview of our runtime framework execution flow

Evaluations

- SET delivers the best throughput, achieving an average of 2.18×, 2.1×, 1.17×, 1.39× speedup against baselines, respectively

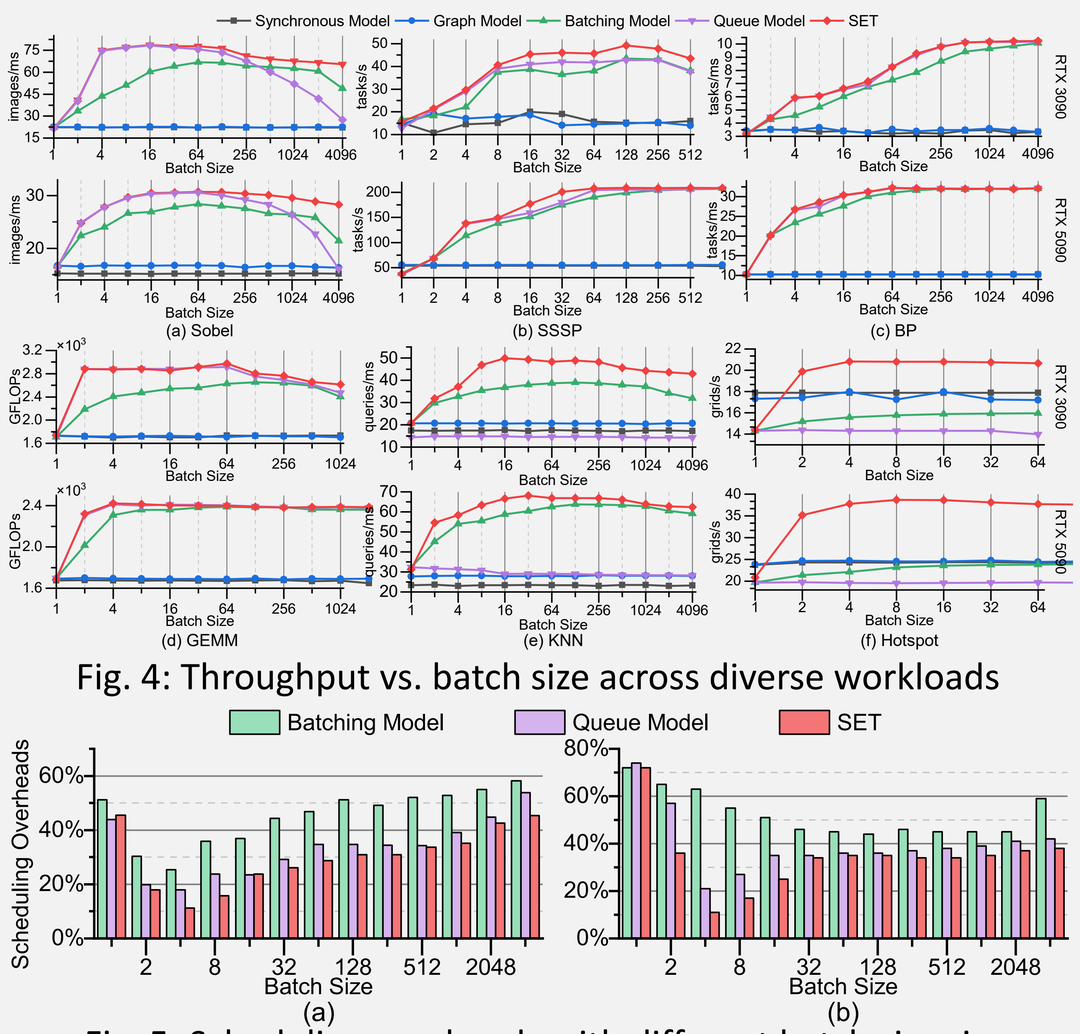

Fig. 4: Throughput vs. batch size across diverse workloads

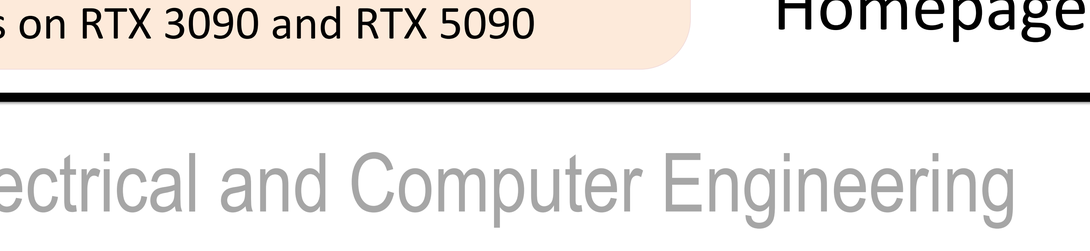

Fig. 5: Scheduling overheads with different batch sizes in different models. (a) on RTX 3090 (b) on RTX 5090

Paper (Arxiv)

Summary:

- We analytically decompose scheduling overheads in CUDA graph pipelines into intra-batch and inter-batch components
- We propose SET, a framework for adaptive task assignment with $O(1)$ synchronization overhead
- We evaluate SET across diverse workloads, achieving 1.15-1.44 × and 18-54% lower scheduling overheads on RTX 3090 and RTX 5090

Homepage