Tensor Program Hyperparameter Search Parallelization with Shared Address Space and Message Passing Models

Kit Ao, Joshua Hong
{kitao, jjhong}@andrew.cmu.edu
Carnegie Mellon University

Introduction

Modern machine learning applications rely on tensor programs that are optimized through engineered rules and heuristics. Building off of recent research involving automated tensor program optimization, we apply Shared Address Space and Message Passing parallelization models to one such optimizer, **Mirage**, by considering potential resource bottlenecks and workload imbalance.

Background

Our project builds off of the Mirage optimizer, which aims to discover and verify possible tensor program optimizations. We focus on the graph generation and probabilistic verification steps, which perform an exhaustive search similar to DFS over the kernel, thread block, and thread levels in order to find configurations that are equivalent to the input tensor program.

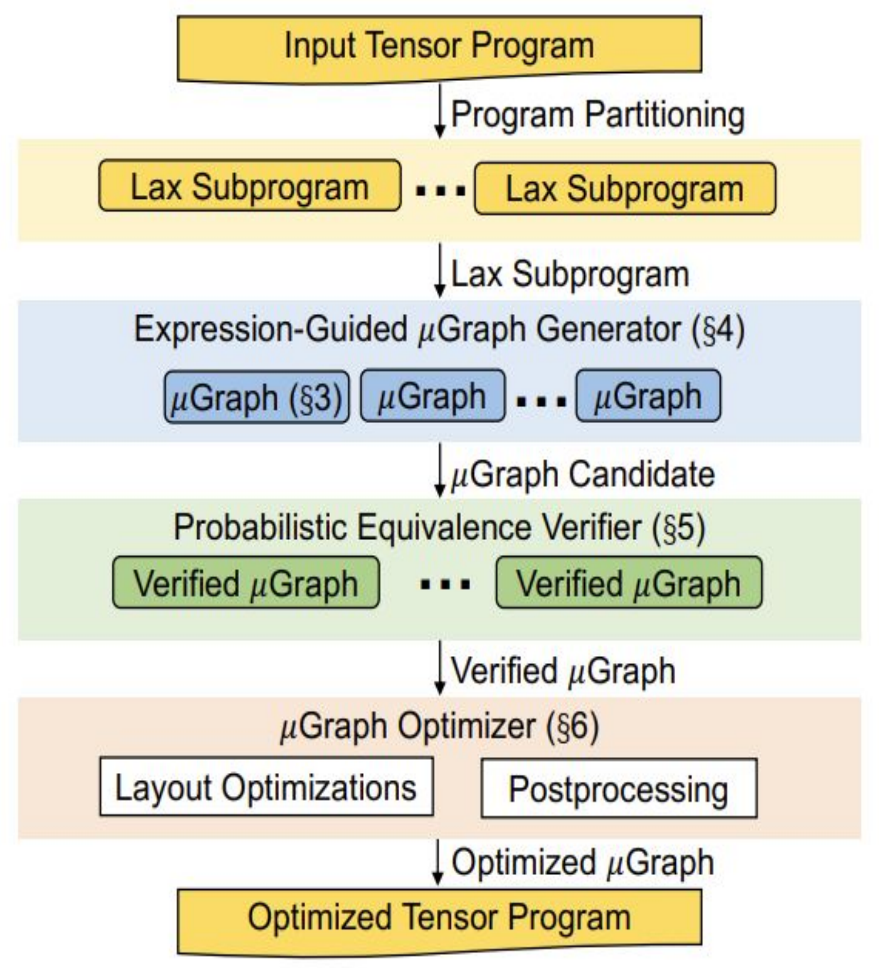
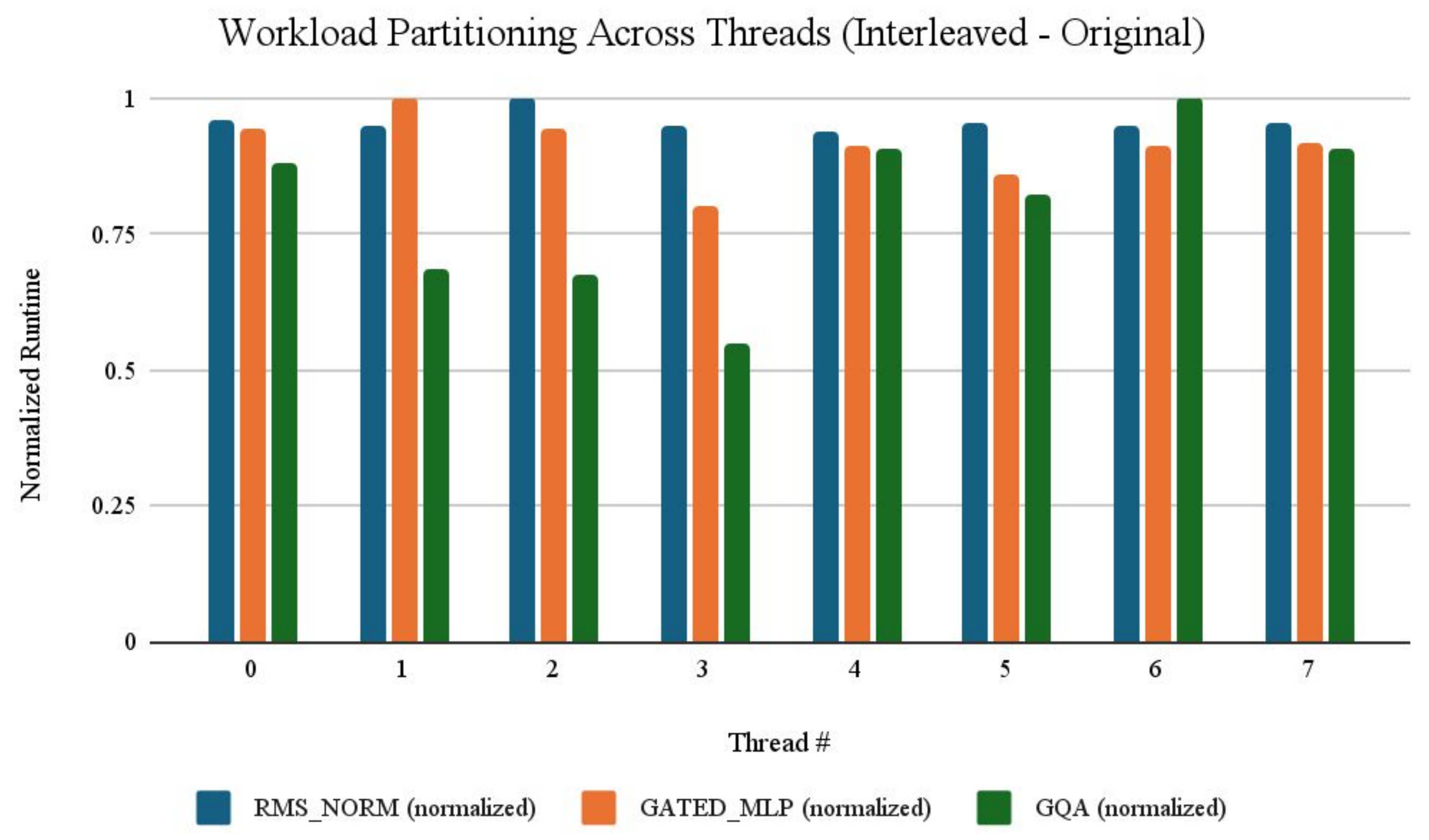


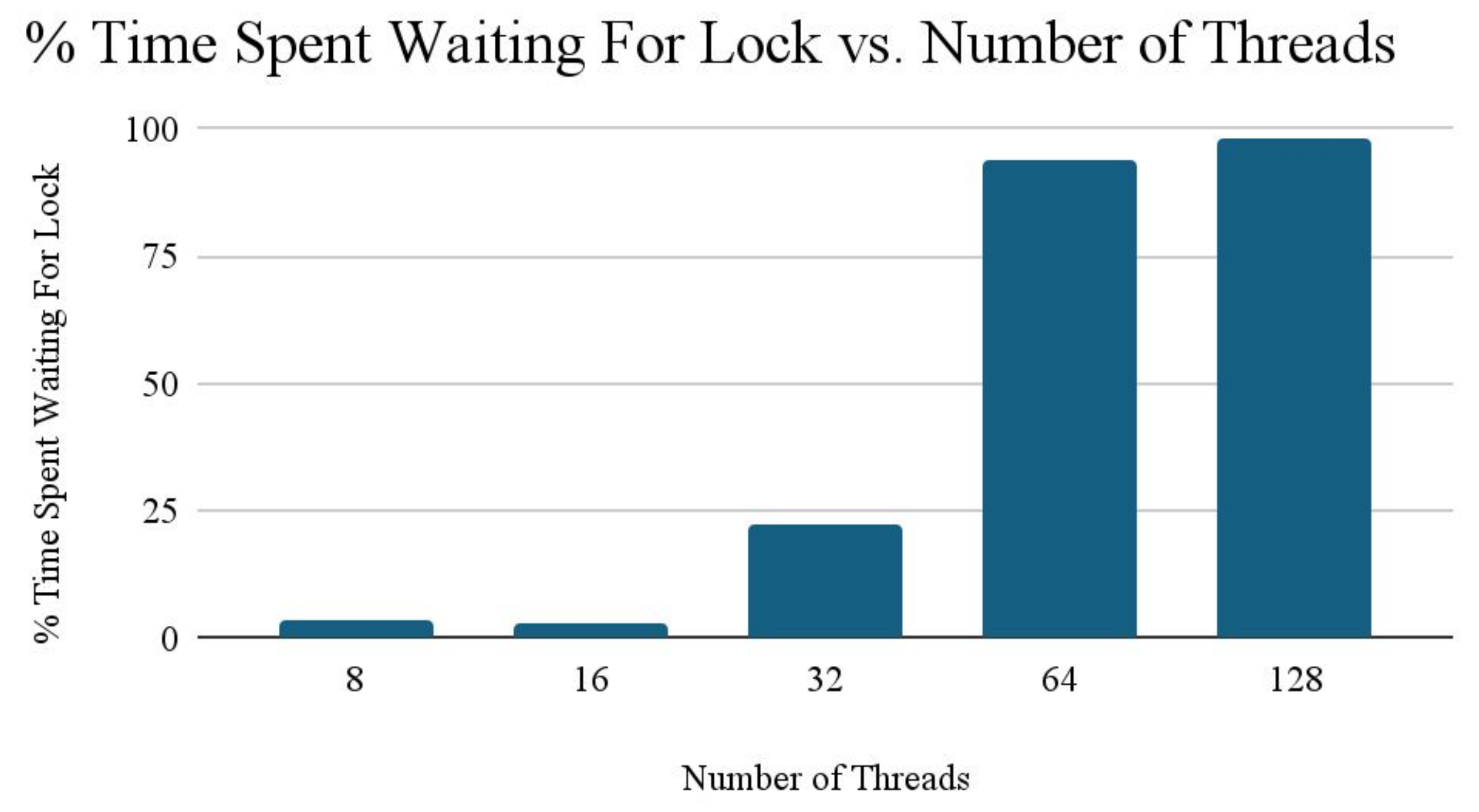
Figure 1. An overview of Mirage.

Challenges

With Mirage originally using a multithreaded approach with interleaved static workload assignment, our preliminary research revealed possible areas of improvement

- Workload Imbalance
  - Early pruning of search nodes can lead to imbalance across threads
- Resource Contention
  - Threads contend when using the Z3 solver and GPU resources for pruning and verification, especially with high thread counts



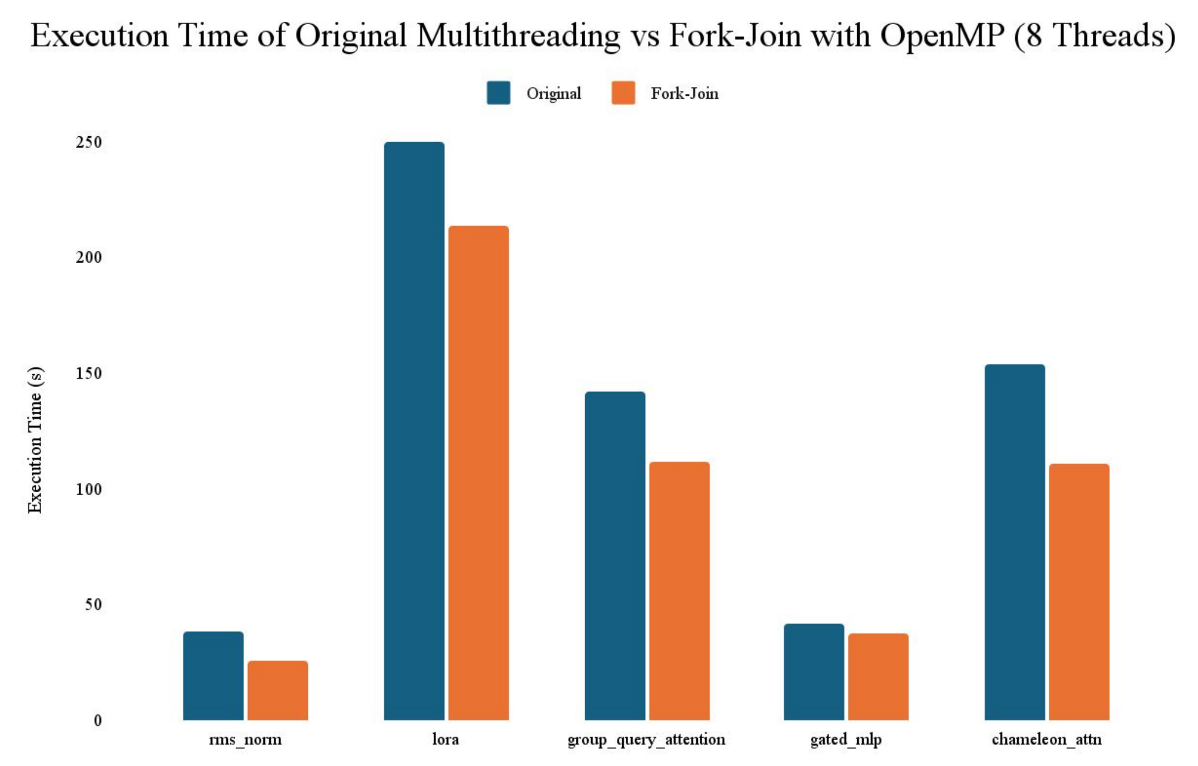


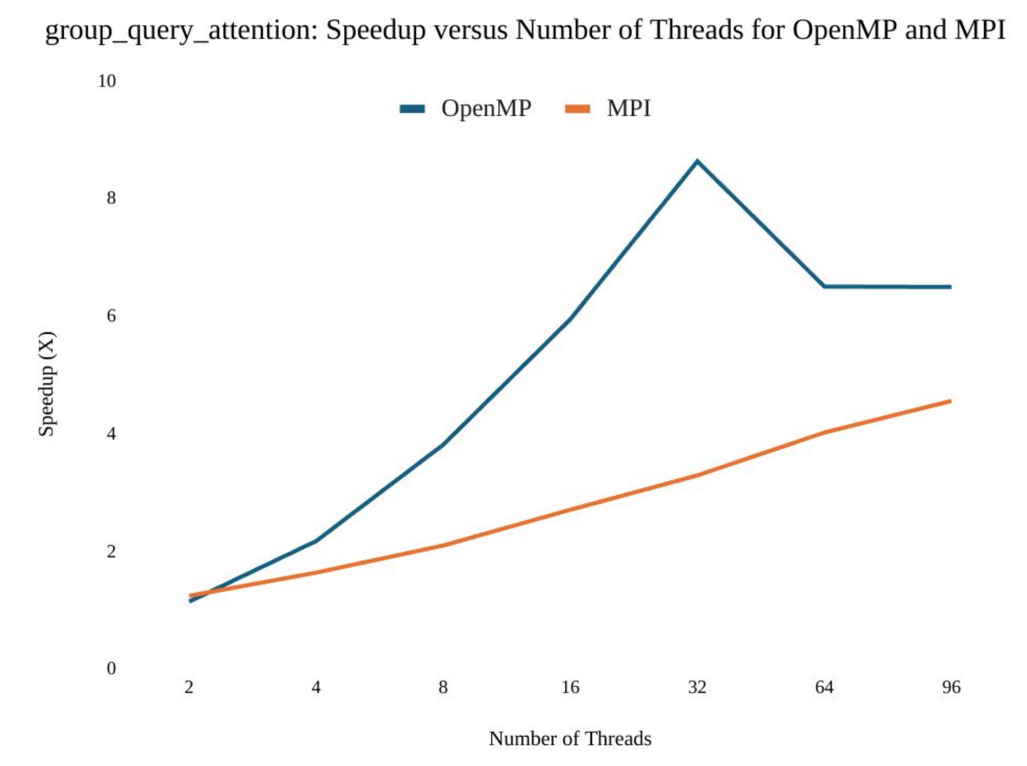
Approach

To address the observed issues with the original parallelization methods, we use OpenMP and MPI to implement shared address space and message passing models respectively

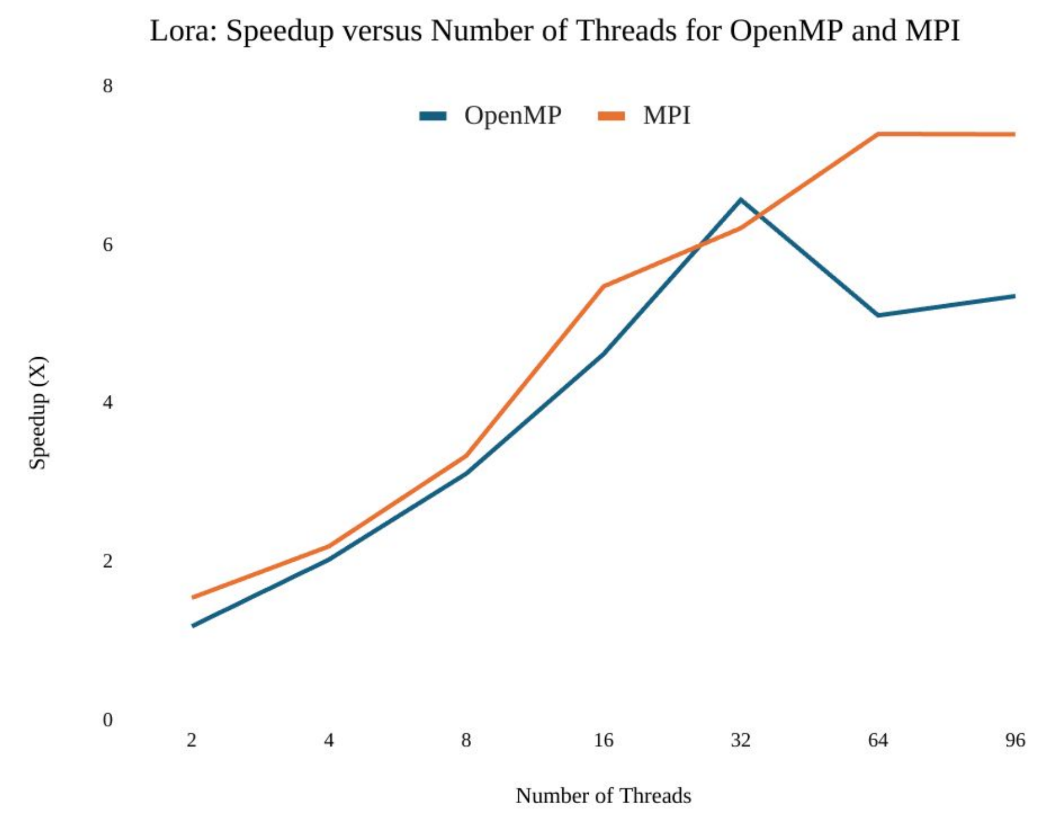
- OpenMP
  - We use OpenMP to achieve dynamic workload assignment, which should allow for balanced workloads across threads
  - Lightweight OpenMP threads allow for parallelization to potentially scale further than the original implementation with pthreads
- MPI
  - We use MPI to distribute computation beyond a single machine, allowing for further parallelism
  - We minimize communication overhead by generating and statically assigning tasks on all nodes and communicating search results to the root node

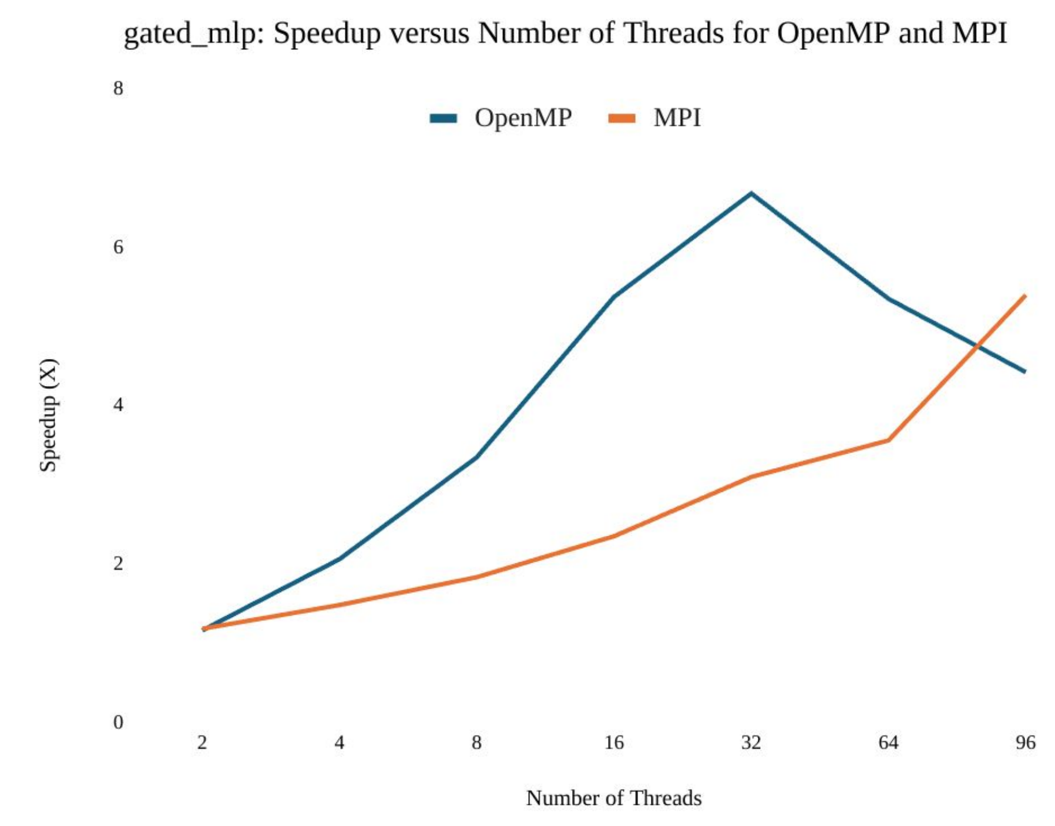
Results and Analysis





- The OpenMP approach showed significant speedup compared to the original implementation, up to 1.50x
  - Speedup is more apparent on tasks with large previously observed workload imbalances
- The MPI approach showed less speedup than the OpenMP approach, but appeared to scale better with high thread count
  - This is potentially due to the static workload distributed used, reintroducing the issue of workload imbalance





Future Work

- Implement dynamic work assignment in the MPI approach to address workload imbalance
- Combine OpenMP and MPI approaches to take advantage of additional parallelism on each node

References

Wu, M., Cheng, X., Padon, O., & Jia, Z. (2024). A Multi-Level Superoptimizer for Tensor Programs. arXiv preprint arXiv:2405.05751.