# Prefix and output-length aware scheduling for efficient online LLM inference

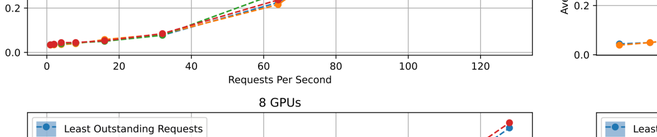
Iñaki Arango,[1,2,3] Ayush Noori,[1,2,3] Yepeng Huang,[3] Rana Shahout,[2] Minlan Yu [3]

[1] Harvard College; [2] Harvard John A. Paulson School of Engineering and Applied Sciences; [3] Harvard Medical School

## Motivation

LLM inference in large data centers can benefit from data parallelization, where models are replicated across GPU devices that can serve requests in parallel. How should we assign requests to GPU workers? Instead of dividing requests evenly and randomly, in real-world applications, requests exhibit patterns that can be exploited to improve performance. These include:

### Shared prefixes

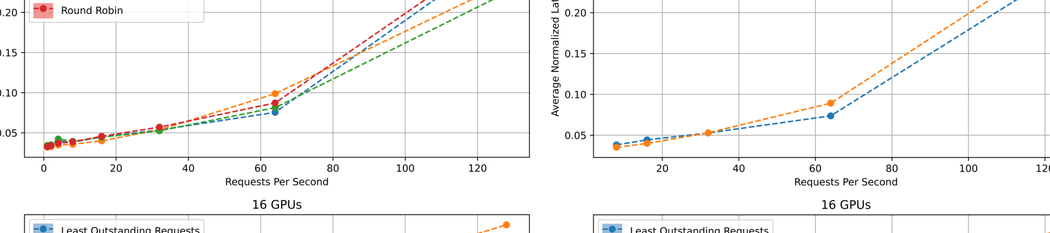



**Applications with sharing**

- General purpose chatbots
- Embedded language interfaces
- Coding
- Document-grounded QA
- Search engines
- Digital workers
- Scientific research

Adapted from Srivatsa *et al.* (2024)

### Variable output length




## Benchmarking PREBLE

The first approach to benefit from prompt sharing under a distributed LLM serving system with data parallelism across multiple GPUs was PREBLE (Srivatsa *et al.*, 2024). However, PREBLE sets the expected decode output length equal to the average output length of requests during scheduling. Therefore, PREBLE may induce imbalanced workloads on different GPUs. Here, we extend PREBLE **by integrating prefix-aware scheduling with output length-aware scheduling** of S$^3$ (Jin *et al.*, 2024). We extend PREBLE's simulator of LLM inference to benchmark PREBLE vs. baseline prefix-unaware schedulers and identify opportunities for improvement. We find that:

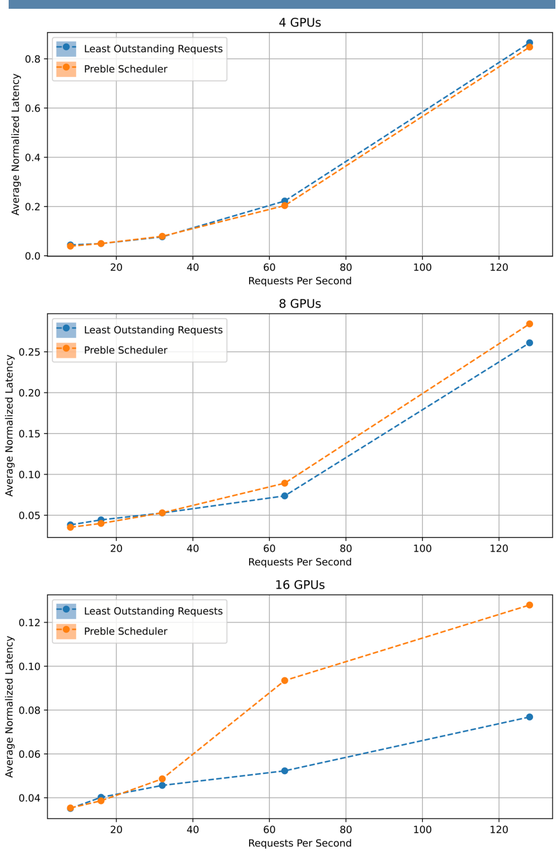
1. **At high request rates, PREBLE is outperformed by several prefix-unaware schedulers.**
2. **This effect is exacerbated by an increase in the number of GPUs available for inference.**
3. **PREBLE suffers from overhead introduced by its E2 scheduler, which scales with both # of GPUs and request rate.**
4. **Decode length heterogeneity worsens PREBLE performance.**

## We evaluated and benchmarked PREBLE across…

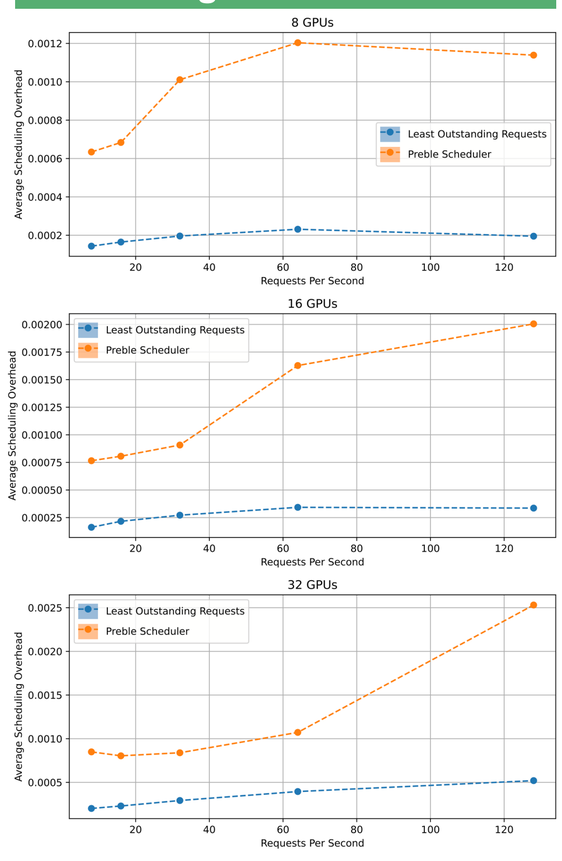
### Request rate



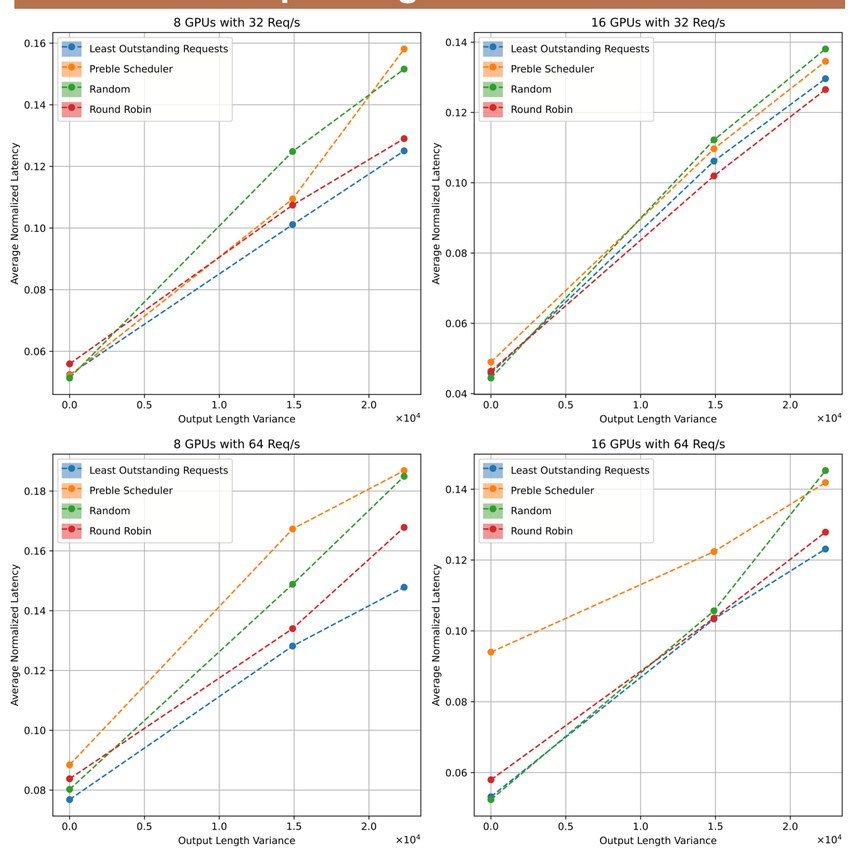

### GPU count



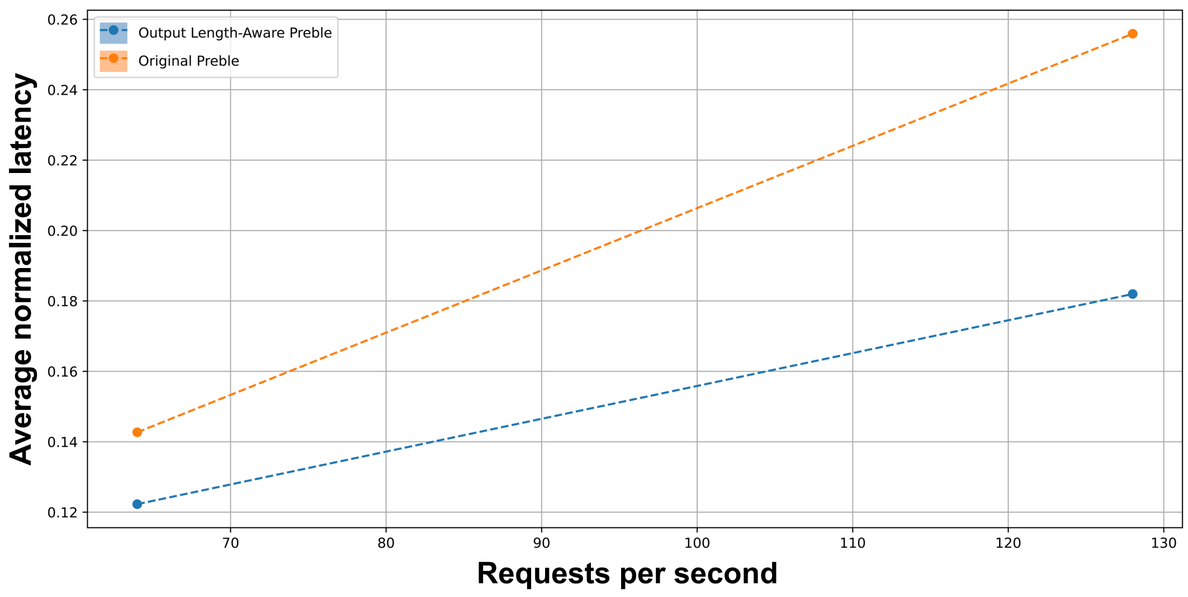

### Scheduling overhead




### Variance in output length




## Adding output-length aware scheduling to PREBLE




Having carefully characterized the scalability challenges associated with PREBLE, we sought to improve its performance by leveraging both prefix-aware and output length-aware scheduling. We build on the E2 scheduler of PREBLE by considering prefix sharing, fairness, and output length. As a proof-of-concept, we use a perfect oracle of true output length. However, decode length **can also be predicted**.

To incorporate output length, we modify the global prefix tree of the E2 scheduler. We evaluate this modified version of PREBLE on a dataset with high variance in token lengths, created using our **ReAct-based variance-customizable dataset generator**.

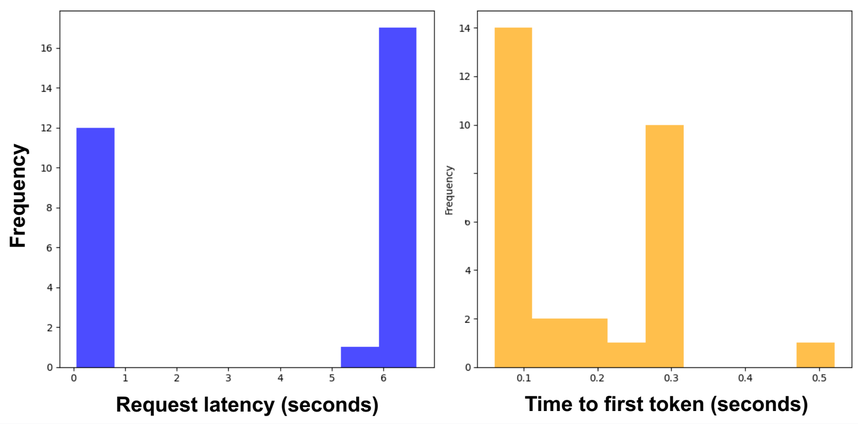
**When using output length for per-GPU load calculation, we improve the performance of PREBLE in high-demand settings, with 14.31% reduced latency at 64 RPS (0.1223 vs. 0.1427) and 28.89% reduced latency at 128 RPS (0.1820 vs. 0.2559).**

### Predicting decode length

We trained a lightweight 6-layer BERT-based `all-MiniLM-L6-v2` language model to predict decode length on the `Alpaca-52K` dataset, achieving 4.8× performance over random.

| Metric | Score | Metric | Score |
|---|---|---|---|
| Accuracy | 0.24 | Recall | 0.24 |
| $F_1$ score | 0.22 | AUROC | 0.85 |
| Precision | 0.23 | MCC | 0.20 |

### Generating a variance-maximizing benchmark




To understand how decode length heterogeneity impacts PREBLE, we generated artificial benchmarking datasets from ReAct whose output lengths can follow any discrete distribution, then used this to create a benchmark with high variance in token lengths (1 vs. 300 tokens).

inaki.io www.ayushnoori.com
{inakiarango, anoori}@college.harvard.edu

R.S. and M.Y. are partially supported by NSF CNS NeTS 2107078. This work was supported in part by ACE, one of the seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

**Harvard** John A. Paulson **School of Engineering** and Applied Sciences