# Parsimonious Learning-Augmented Caching

Sungjin Im [1] Ravi Kumar [2] Aditya Petety [1] Manish Purohit [2]

[1] UC Merced

[2] Google

# Problem Definition

**Online Caching**

- Sequence of pages $\Gamma = \langle p_1, p_2, \dots \rangle$, $p_i \in \mathcal{U}$ arrives online

# Problem Definition

**Online Caching**

- Sequence of pages $\Gamma = \langle p_1, p_2, \ldots \rangle$, $p_i \in \mathcal{U}$ arrives online
- Cache can not store more than $k$ pages at any time, pay 1 per cache miss

# Problem Definition

**Online Caching**

- Sequence of pages $\Gamma = \langle p_1, p_2, \dots \rangle$, $p_i \in \mathcal{U}$ arrives online
- Cache can not store more than $k$ pages at any time, pay 1 per cache miss

**Learning-augmented setting:** Algorithm receives a prediction for the next arrival time of the requested page

# Problem Definition

**Online Caching**

- Sequence of pages $\Gamma = \langle p_1, p_2, \ldots \rangle$, $p_i \in \mathcal{U}$ arrives online
- Cache can not store more than $k$ pages at any time, pay 1 per cache miss

**Learning-augmented setting:** Algorithm receives a prediction for the next arrival time of the requested page

**Goal:** Minimize the number of cache misses

# Motivation

**Why Learning-augmented algorithms?**

- Traditional algorithms provide guarantees over *all* inputs

# Motivation

### Why Learning-augmented algorithms?

- Traditional algorithms provide guarantees over *all* inputs
- Learning-augmented algorithms use predictions to improve performance

# Motivation

**Why Learning-augmented algorithms?**

- Traditional algorithms provide guarantees over *all* inputs
- Learning-augmented algorithms use predictions to improve performance
- Also retains *worst-case* guarantees

# Motivation

## Why Learning-augmented algorithms?

- Traditional algorithms provide guarantees over *all* inputs
- Learning-augmented algorithms use predictions to improve performance
- Also retains *worst-case* guarantees

## Parsimonious

- Obtaining predictions is computationally expensive
- Desirable to use predictions *parsimoniously*

# Our Contributions

- Allow the algorithm to query $b$ pages per cache miss

# Our Contributions

- Allow the algorithm to query $b$ pages per cache miss
- Develop an algorithm that achieves competitive ratio of
  - $O(\log_{b+1} k)$ when predictions are good
  - $O(\log k)$ even when predictions are bad

# Our Contributions

- Allow the algorithm to query $b$ pages per cache miss
- Develop an algorithm that achieves competitive ratio of
  - $O(\log_{b+1} k)$ when predictions are good
  - $O(\log k)$ even when predictions are bad
- Our algorithm achieves a near-optimal trade-off between the competitive ratio and no. of queries per cache miss

# Our Contributions

- Allow the algorithm to query $b$ pages per cache miss
- Develop an algorithm that achieves competitive ratio of
  - $O(\log_{b+1} k)$ when predictions are good
  - $O(\log k)$ even when predictions are bad
- Our algorithm achieves a near-optimal trade-off between the competitive ratio and no. of queries per cache miss
- Experimentally show that making around 10% queries
  - improves over traditional algorithms
  - match previous learning-augmented algorithms

# Main Results

### Adaptive query eviction algorithm

- Based on randomized marking algorithm
- Query $b$ unmarked pages
- Evict page with furthest predicted next arrival

### Theorem

*For any integer $b > 0$, there is an $O(\min\{\log_{b+1} k + \mathbb{E}[\eta]/opt, \log k\})$-competitive algorithm for caching that makes at most $b$ queries per cache miss.*

# Main Results

### High level idea

- If predictions are correct, the evicted page will not arrive in the next $k/(b+1)$ time steps (in expectation)

# Main Results

**High level idea**

- If predictions are correct, the evicted page will not arrive in the next $k/(b+1)$ time steps (in expectation)
- Switch to randomized marking when the algorithm makes too many mistakes

# Main Results

### High level idea

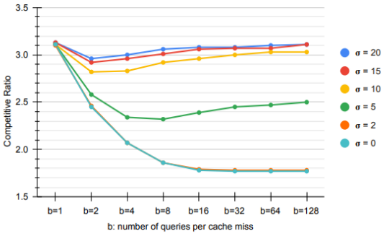
- If predictions are correct, the evicted page will not arrive in the next $k/(b+1)$ time steps (in expectation)
- Switch to randomized marking when the algorithm makes too many mistakes
- Ensures competitive ratio does not exceed $O(\log k)$

# Experiments

## Datasets

- CitiBike 2018
- Sequence length: 25000, cache size: 500

**Predictions:** Ground truth + lognormal error




| Algorithms | Mean Predictions | Synthetic Predictions $\sigma = 0$ | $\sigma = 2$ | $\sigma = 4$ | $\sigma = 6$ |
|---|---|---|---|---|---|
| RandomMarker | 3.14 | 3.14 | 3.14 | 3.14 | 3.14 |
| LRU | 2.86 | 2.86 | 2.86 | 2.86 | 2.86 |
| BlindOracle | 1.92 | 1.00 | 1.02 | 3.92 | 4.15 |
| LVMarker | 2.49 | 1.77 | 1.81 | 2.94 | 3.11 |
| RohatgiMarker | 2.54 | 1.77 | 1.83 | 3.15 | 3.29 |
| RobustOracle | 4.29 | 1.80 | 1.83 | 4.48 | 4.51 |
| AdaptiveQuery-2 | 2.91 | 2.46 | 2.46 | 2.52 | 2.65 |
| AdaptiveQuery-4 | 2.71 | 2.07 | 2.07 | 2.20 | 2.49 |
| AdaptiveQuery-8 | 2.59 | 1.86 | 1.86 | 2.07 | 2.54 |