Applying Online and Reinforcement Learning Techniques to Improve Cache Performance

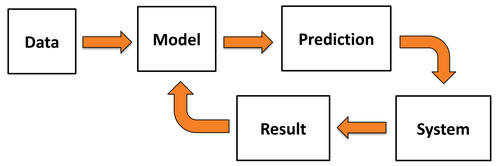
Aninda Manocha, *Princeton University Department of Computer Science*

ABSTRACT

- To address the end of Moore's Law, the development of specialized hardware and parallelism has greatly improved computational performance and efficiency, but data supply remains a bottleneck for many important applications
  - Memory accesses can be orders of magnitude slower than computations
- Caches (local memory hierarchies), are employed to store data that is frequently accessed closer to where computation takes place
  - The latency of a memory access can be significantly reduced
- Different applications have different memory access patterns, which can affect data supply bottlenecks
  - Parallel applications usually contain regular and predictable memory accesses
  - Graph applications are notorious for irregular memory accesses that can lead to more cache misses, which are costly in terms of latency
- Fetching cache data ahead of time can reduce the number of cache misses and consequently improve cache performance and memory bottlenecks
- This project presents online and reinforcement learning techniques that can predict ahead of time if given memory accesses will hit or miss in the cache and prefetch data accordingly

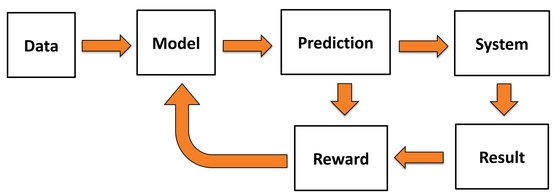
BACKGROUND AND APPROACH

Online machine learning algorithms are useful for incorporating prediction into an evolving system



Online machine learning

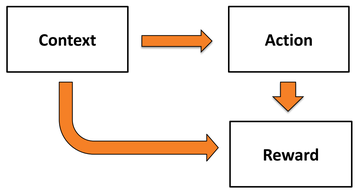
Reinforcement learning is a form of online machine learning that generates a reward per prediction



Reinforcement learning

Contextual bandits

- Multi-armed bandits scenario = aim to allocate a fixed and limited set of resources between competing choices ("arms") to maximize their expected rewards
- Contextual bandits = adapting multi-armed bandits policies to online contextual bandits scenario
  - Use information about state of environment (context) to help make prediction
  - Binary rewards
  - Binary classification algorithms utilized as black-box oracles
- In this project, 2 "arms": a cache hit and a cache miss
  - Reward = whether or not the cache access prediction was correct



Contextual bandits model

Methodology

- *Data preprocessing:* divide address space and label each address based on the division it falls in
  - Perform one-hot encoding of address division labels to create several binary features
- *Cache access prediction:* use model to make a prediction about a memory access (hit/miss)
  - Models are trained over time with accuracies (reinforcement learning models) or true results (classifiers)
- *Prediction-based prefetching:* predict cache hit/miss of future memory access and prefetch data in advance if a miss is predicted
  - If a miss is correctly predicted, then *x* cycles are saved, where *x* is how many memory accesses ahead the prediction is made
  - If a miss is not predicted correctly, then the cache access was actually a hit and the fetched data can be ignored (data is fetched in parallel with the execution of the cache's operations)

CACHE ACCESS PREDICTION

Evaluating cache access prediction:

The datasets are memory traces (first 3000 accesses) of programs:

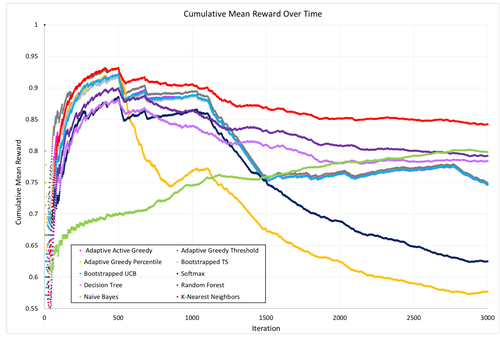
1. *Graph projections* run on a crime dataset
2. A benchmark program that involves many *random memory accesses* in addition to *pointer chasing*

Models:

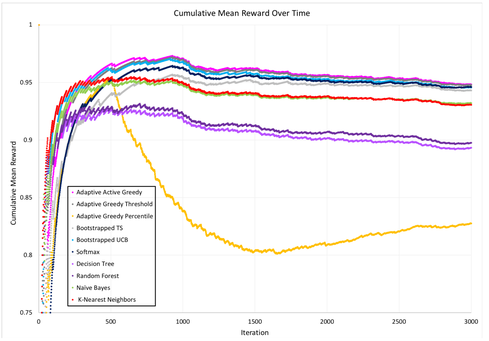
- 6 algorithmic variants of *contextual bandits* (reinforcement learning)
- 4 different *classifiers* (online learning)

Performance:

1. Graph projections (more regular accesses): all classifiers outperform all reinforcement learning models
2. Benchmark: classifiers always outperformed by most reinforcement learning models

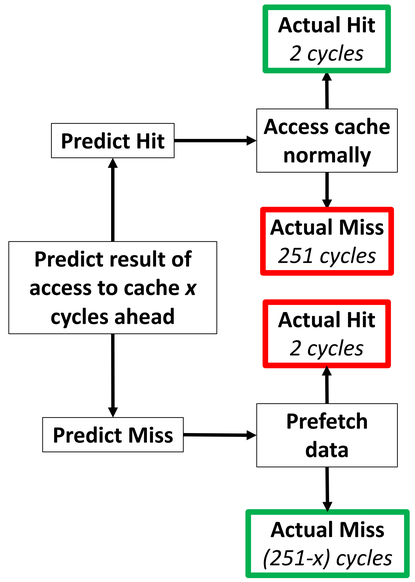


Cumulative accuracy of cache access predictions for a graph algorithm run on a crime dataset.



Cumulative accuracy of cache access predictions for a random accesses and pointer chasing benchmark.

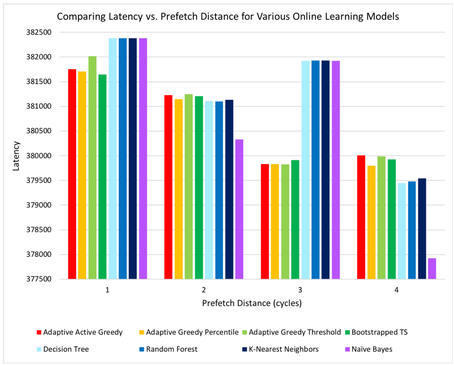
PREDICTION-BASED PREFETCHING



Prediction-based prefetching algorithm

Cache access latencies

- Time to check/search the cache for the existence of an address = 1 cycle
- Hit time = 1 cycle
- Miss time = 250 cycles
- These latencies are based off typical L1 cache latencies
- The values of these latencies can affect the amounts of speedups achieved



Increasing the prefetch distance *x* generally decreases the latency for each model, but there are a few exceptions with distances 3 and 4.

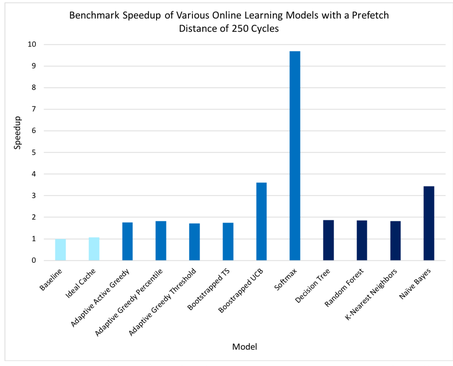
Evaluating prediction-based prefetching

Prefetch distance:

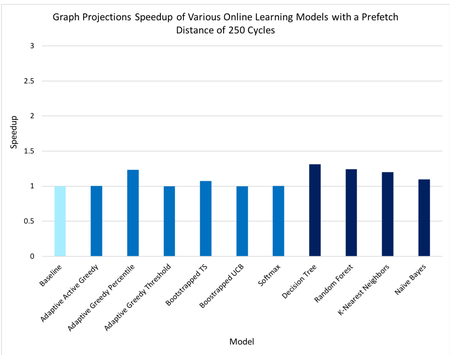
- *x* = prefetch distance (how many cycles ahead of the memory access the prediction is made)
- The greater the prefetch distance, the greater the number of cycles saved on a correctly predicted cache miss (*miss_time* + 1 – *x*)
  - There are more cycles at the beginning which cannot be predicted
- Improved performance results from a greater number of correctly predicted cache misses
  - Performance cannot get worse

Performance:

- Increasing prefetch distance generally improves latency
- Increasing prefetch distance to match the miss time creates speedups for some models



Prediction-based prefetching generates speedups for all models for a benchmark with random + irregular accesses.



Prediction-based prefetching generates speedups for many models for graph projections with a smaller cache.

ACKNOWLEDGEMENTS

This project is an informative component of DECADES, a larger research project I am contributing towards. I would like to thank my advisor, Professor Margaret Martonosi, and my research collaborators, Tyler Sorensen and Esin Tureci, for their advice and inspiration.

I would also like to thank my research collaborator, Luwa Matthews, for his work on the DECADES simulator which generated the program memory traces that were used to train and test these reinforcement learning techniques.

REFERENCES

1. M. Hashemi, K. Swersky, J. A. Smith, G. Ayers, H. Litz, J. Chang, C. Kozyrakis, and P. Ranganathan. Learning memory access patterns. In *Proceedings of the 36th International Conference on Machine Learning*, 2018.
2. C. Wu, A. Jaleel, W. Hasenplaugh, M. Martonosi, S. C. Steely Jr., and J. Emer. SHiP: Signature-based hit predictor for high performance caching. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, 2011.
3. D. Cortes. Adapting multi-armed bandits policies to contextual bandits scenarios. In *CoRR*, 2018. (Code and examples from https://github.com/david-cortes/contextualbandits)

amanocha@cs.princeton.edu