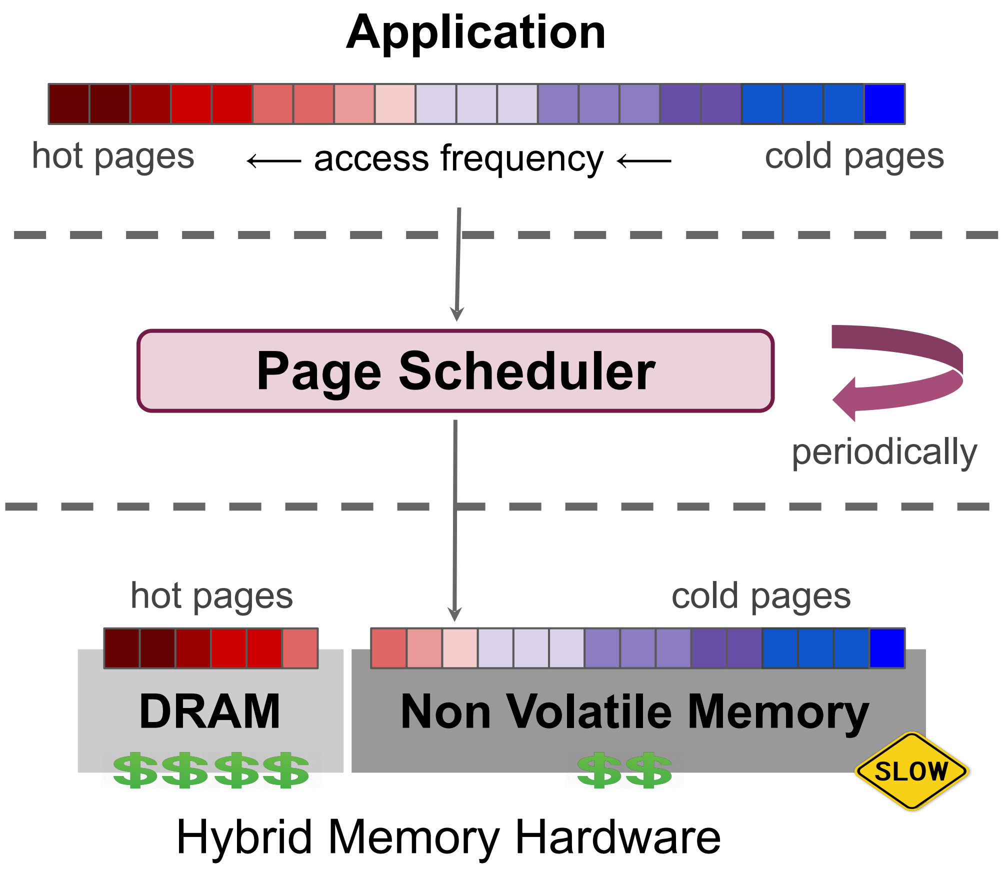
Kleio: A Hybrid Memory Page Scheduler with Machine Intelligence

Thaleia Dimitra Doudali, Sergey Blagodurov, Abhinav Vishnu,
Sudhanva Gurumurthi, Ada Gavrilovska

https://www.cc.gatech.edu/~tdoudali/

Problem Space

Dynamic Data Management in Hybrid Memory Systems



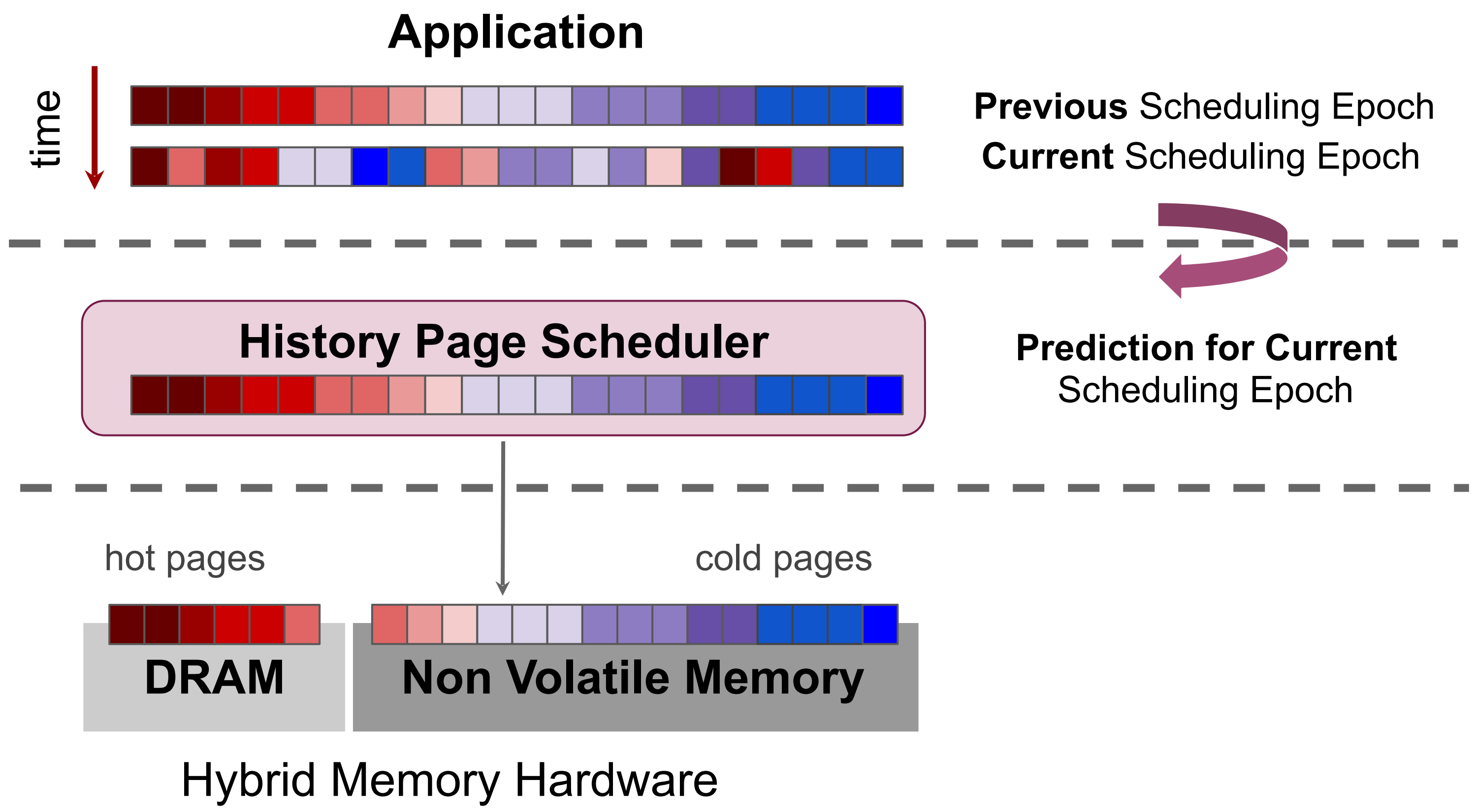
3. Problem
How to predict which data is hot so as to timely migrate it in DRAM.

2. Approach
Timely allocation in DRAM of frequently accessed (hot) data through periodic data migrations can boost application performance.

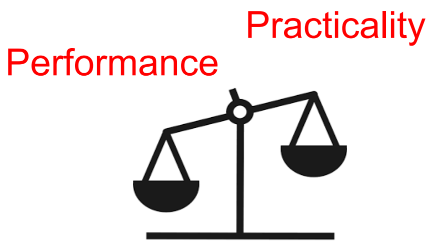
1. Challenge
Use of Non Volatile Memory (NVM) to extend main memory capacity reduces the system cost in return for application performance degradation.

State-of-the-art Solution

Lightweight history-based predictions



A **History[1] page scheduler** periodically migrates data such that DRAM hosts the hot data of the previous scheduling epoch until capacity is full.



[1]**Reference:** *M.R.Meswani,S.Blagodurov,D.Roberts,J.Slice,M.Ignatowski,and G.H.Loh. 2015. Heterogeneous memory architectures: A HW/SW approach for mixing die-stacked and off-package memories. In 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*

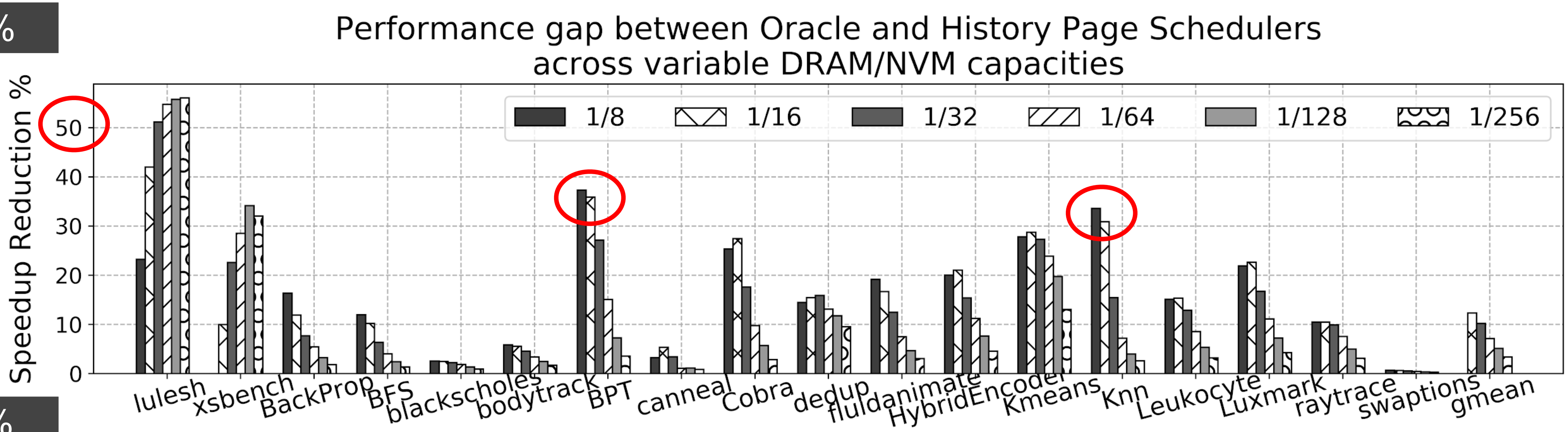
Existing Solutions

Leave a significant gap for possible performance improvements

History = x%

The higher
The worse

Oracle = 0%

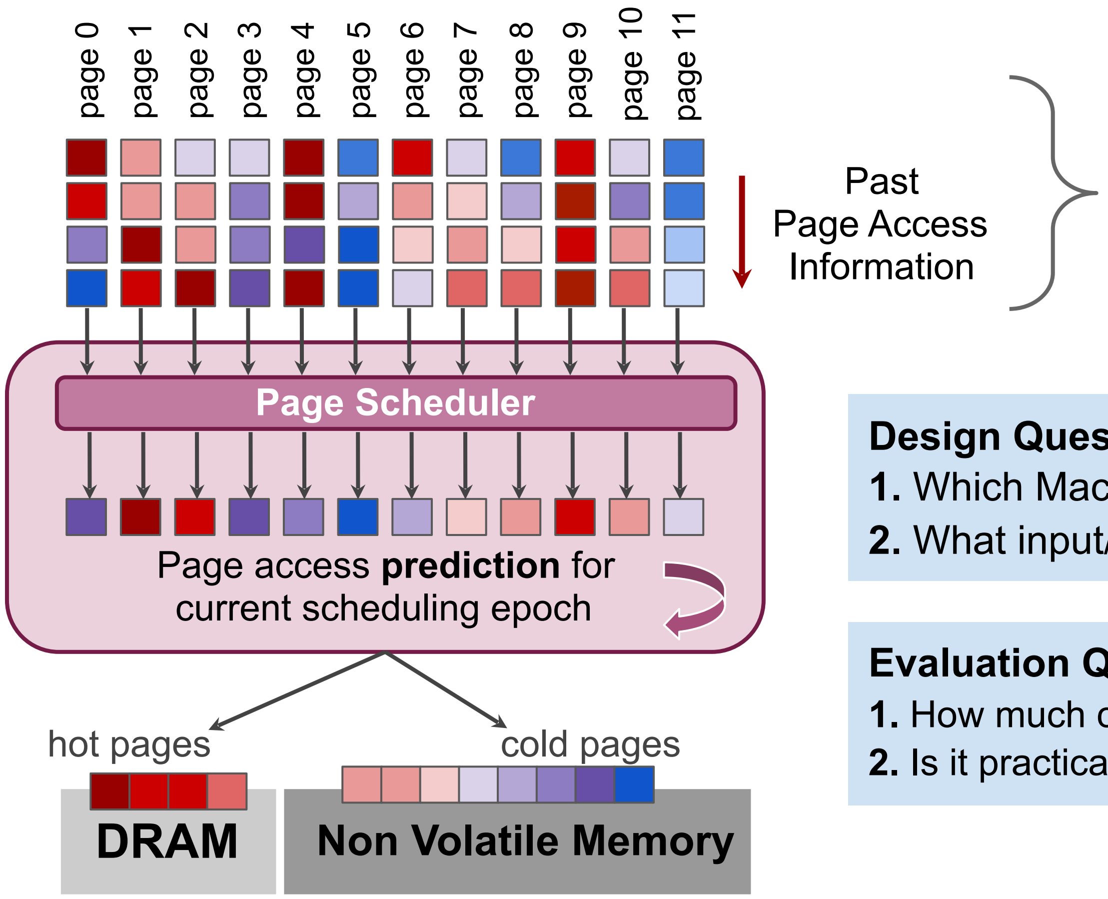


Added Performance Reduction due to Page Scheduling

Simple history-based *page scheduling methods may end up causing significant* *<u>additional</u>* *performance degradation in applications executing over hybrid memory systems.*
We need something more clever to close the gap!

Solution Design

Questions that need to be answered



How can we use **Machine Intelligence** in order to combine *past* access information into an *accurate prediction* of *future* behavior?

Design Questions:

1. Which Machine Intelligence (MI) method to use?

2. What input/output fits the page scheduling description?

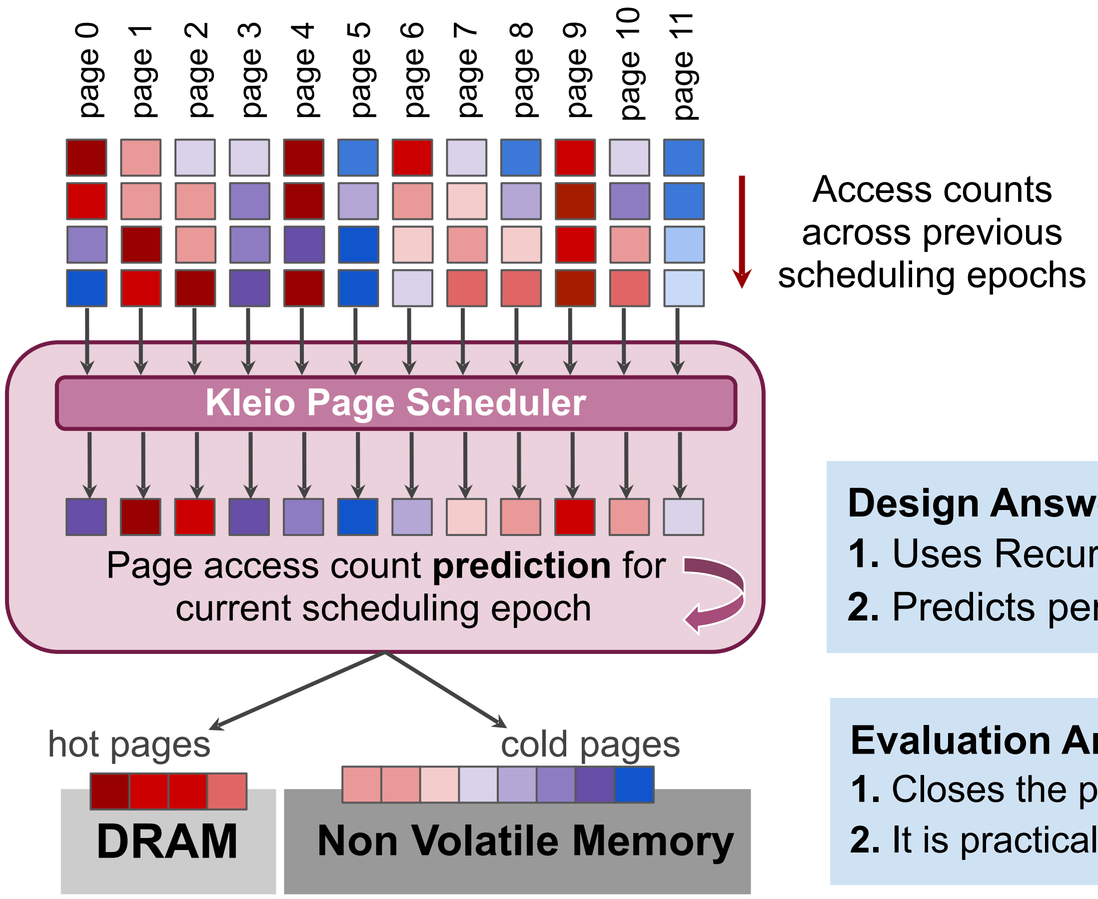
Evaluation Questions:

1. How much can it reduce the performance gap? How accurate are the predictions?

2. Is it practical to integrate into future systems?

Solution Overview

Kleio Page Scheduler answers all the questions



Kleio* is a machine intelligent page scheduler for hybrid memory systems.

*According to the ancient Greek mythology, Kleio was the muse of history, daughter of Mnemosyne, goddess of memory.

Design Answers:

1. Uses Recurrent Neural Networks (RNNs).

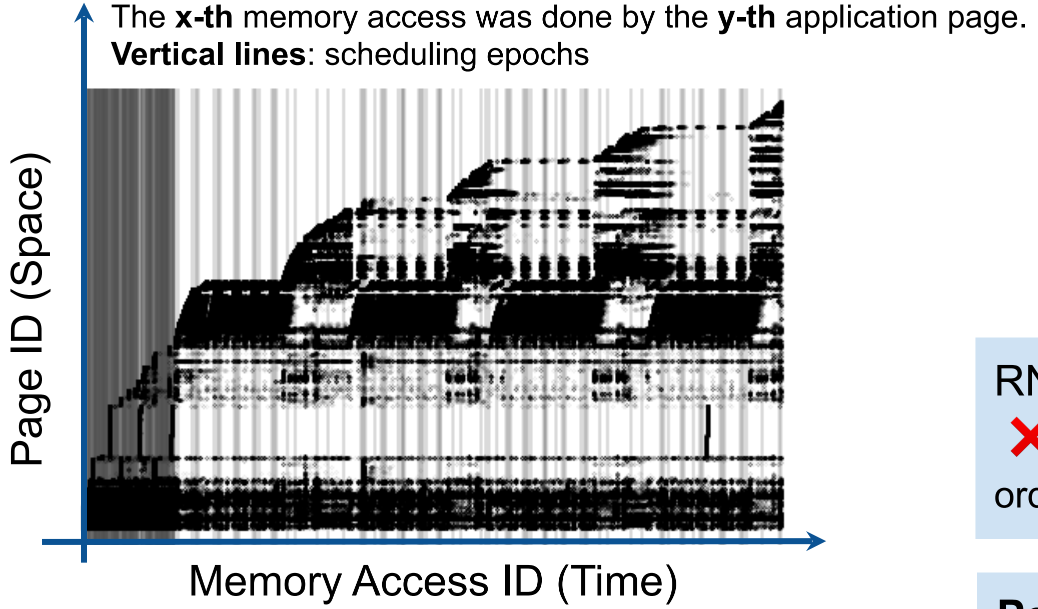
2. Predicts per page access counts.

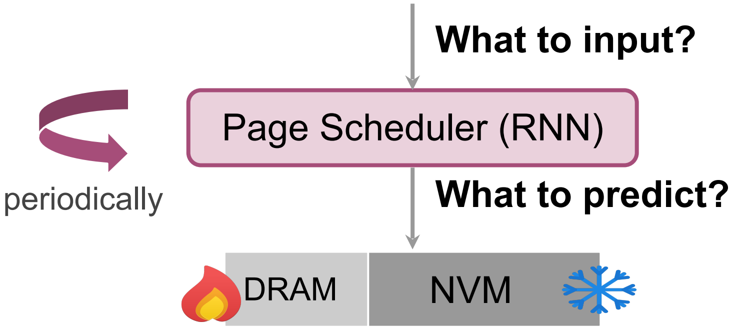
Evaluation Answers:

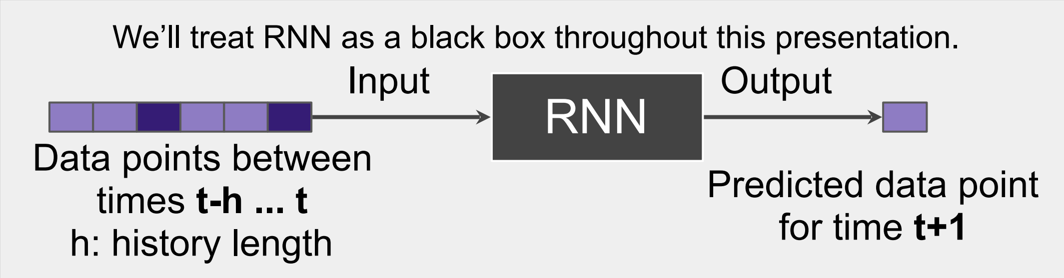
1. Closes the performance gap by **80%.**

2. It is practical since it identifies the page subset that needs MI-based management.

Solution Design
Suitable RNN Input Format







RNNs as used in **Prefetching** [1]**:** Which page will be accessed next?

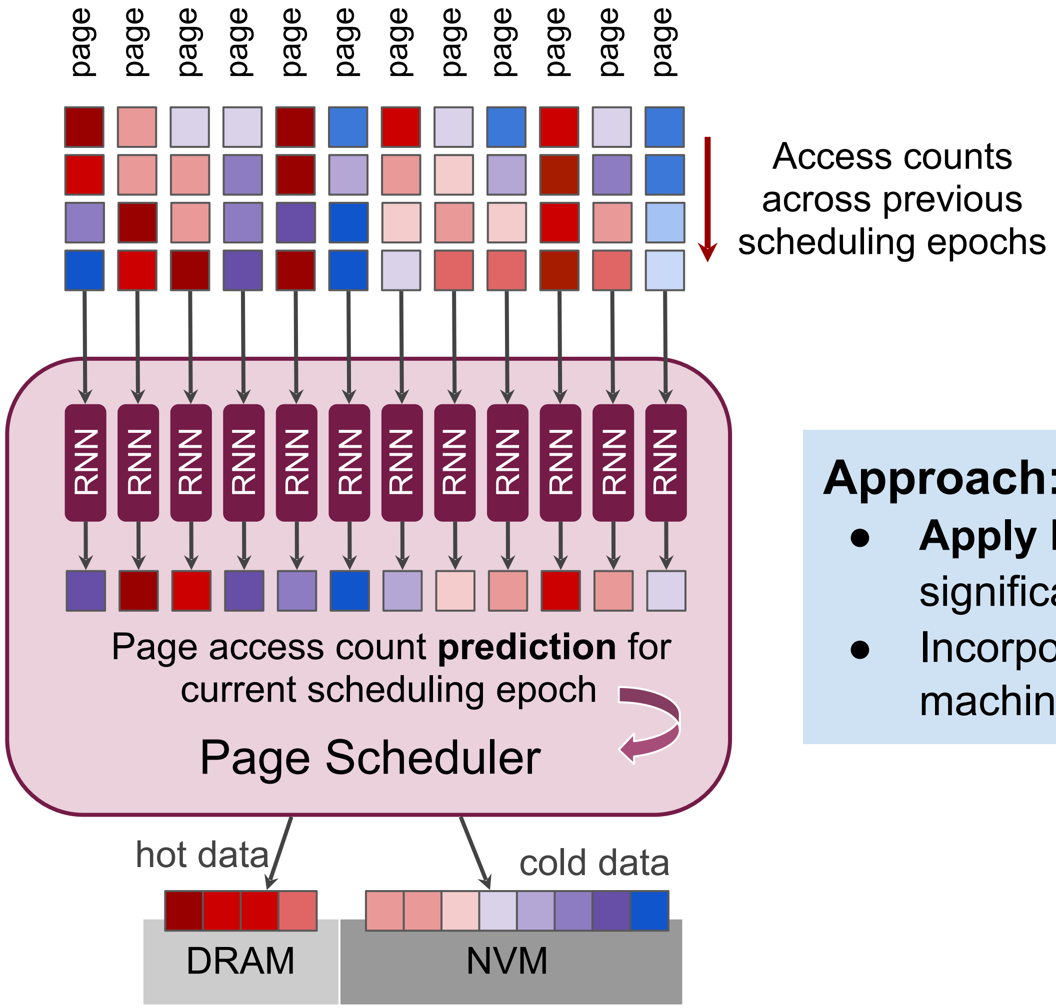
❌ Not suitable due to high training overheads and low accuracy levels in order to make a decision for all pages.

Per Page Prediction: How many times a page was accessed.

✅ Suitable to deliver low training times and adequate prediction accuracy.

[1] **Reference:** Hashemi, M., Swersky, K., Smith, J., Ayers, G., Litz, H., Chang, J., Kozyrakis, C. & Ranganathan, P.. (2018). Learning Memory Access Patterns. Proceedings of the 35th International Conference on Machine Learning, in PMLR 80:1919-1928

Solution Design
Per Page Prediction

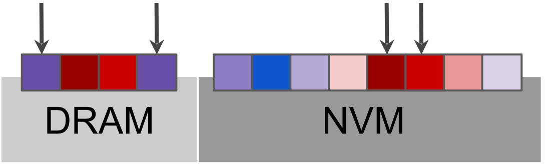


Not really scalable..
HPC and Big Data applications can have millions of pages!

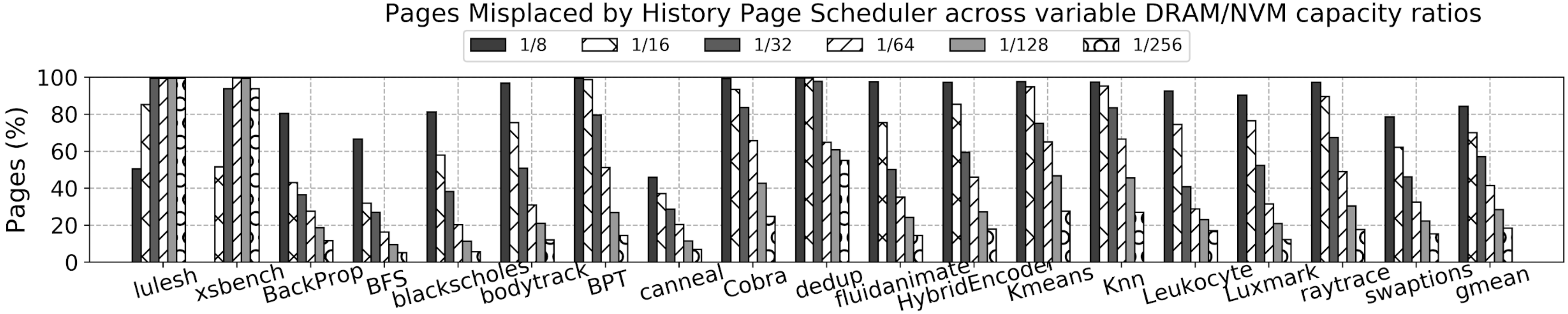
Approach:
- **Apply RNNs** on the **page subset** whose timely DRAM allocation brings significant performance improvement.
- Incorporate **lightweight current state-of-the art** solutions without machine intelligence for the **remaining pages**.

Solution Design

Pages not misplaced by the History page scheduler don't need Machine Intelligence



A page is **misplaced** when at the start of a scheduling epoch it is not allocated in DRAM, even though it was hot, because the scheduler mispredicted its high access frequency.

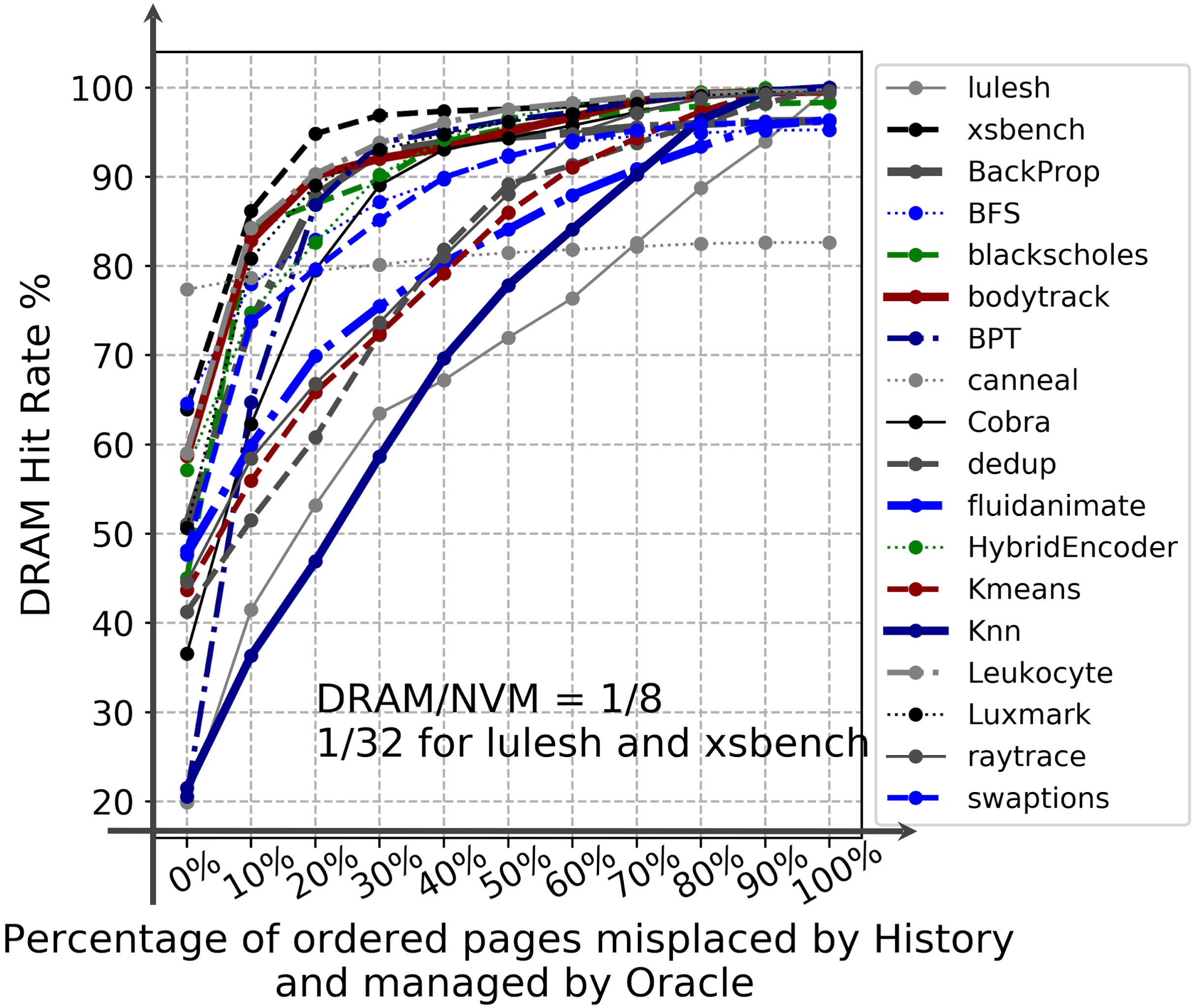


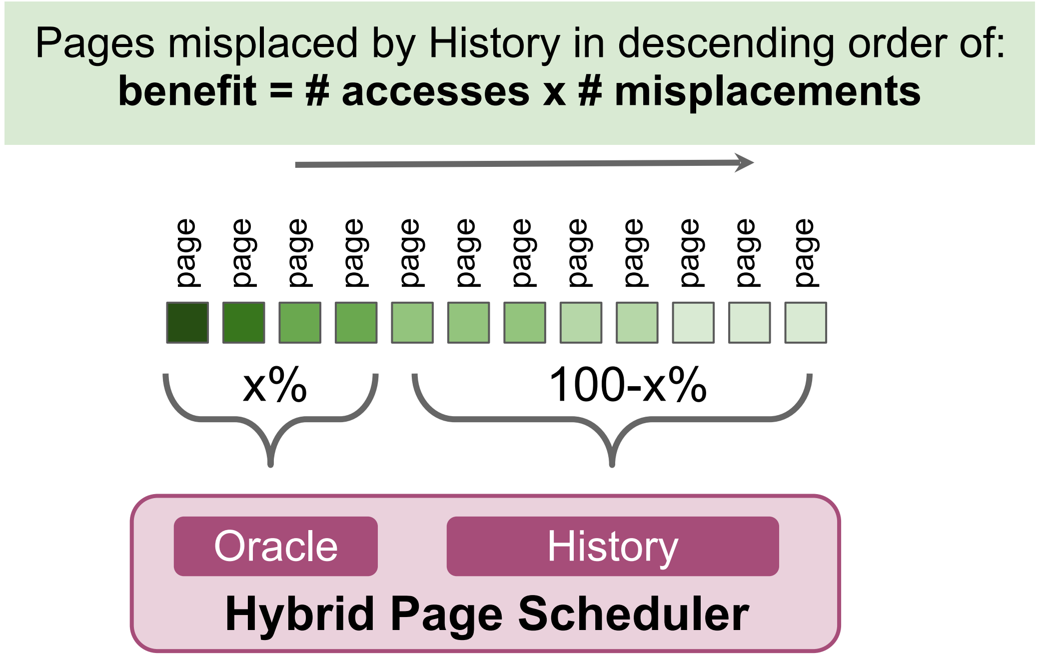
The History page scheduler reduces the number of pages we need to manage more cleverly.
Still, the number be significant especially for DRAM:NVM capacity ratios that are expected in future systems, such as 1/8.

Can we further reduce the number of pages that need more intelligent management?

Solution Design

Prioritize for RNNs the misplaced pages that are highly accessed



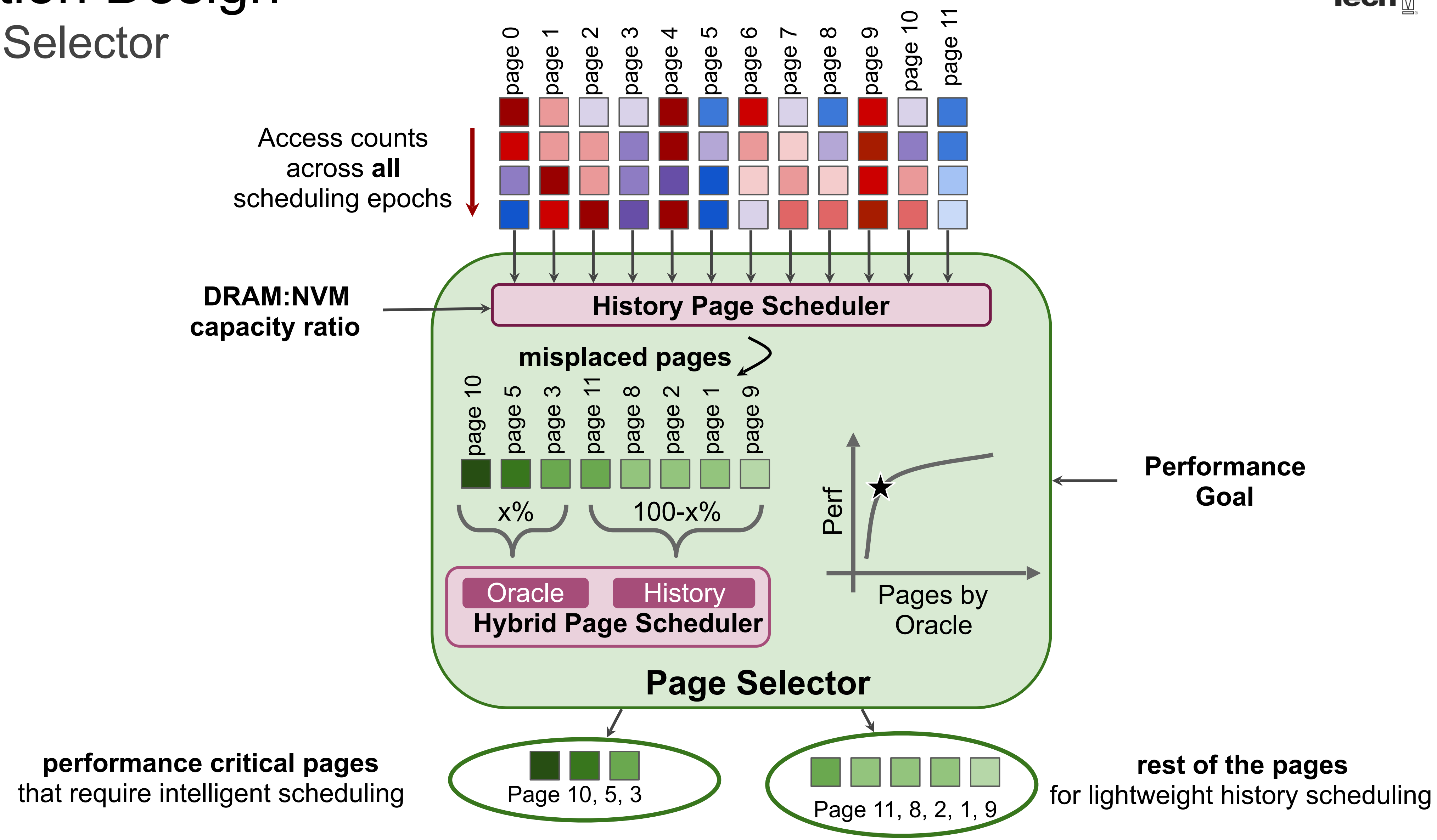


Question: How does performance increase, the more pages we manage intelligently via Oracle?

Answer: Non linearly. Only a small page subset with high benefit needs intelligent management.

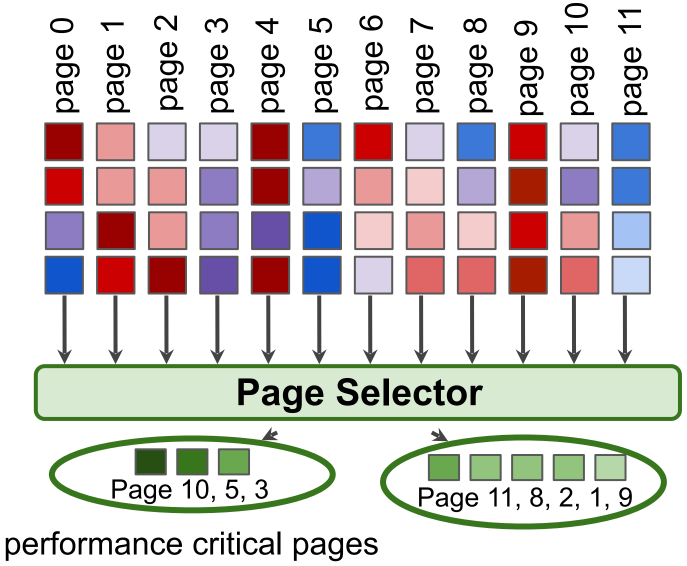
Solution Design

Page Selector



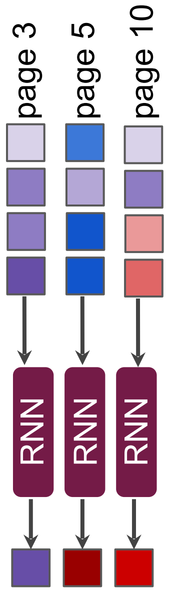
Solution Overview

Step 1: Page Selection



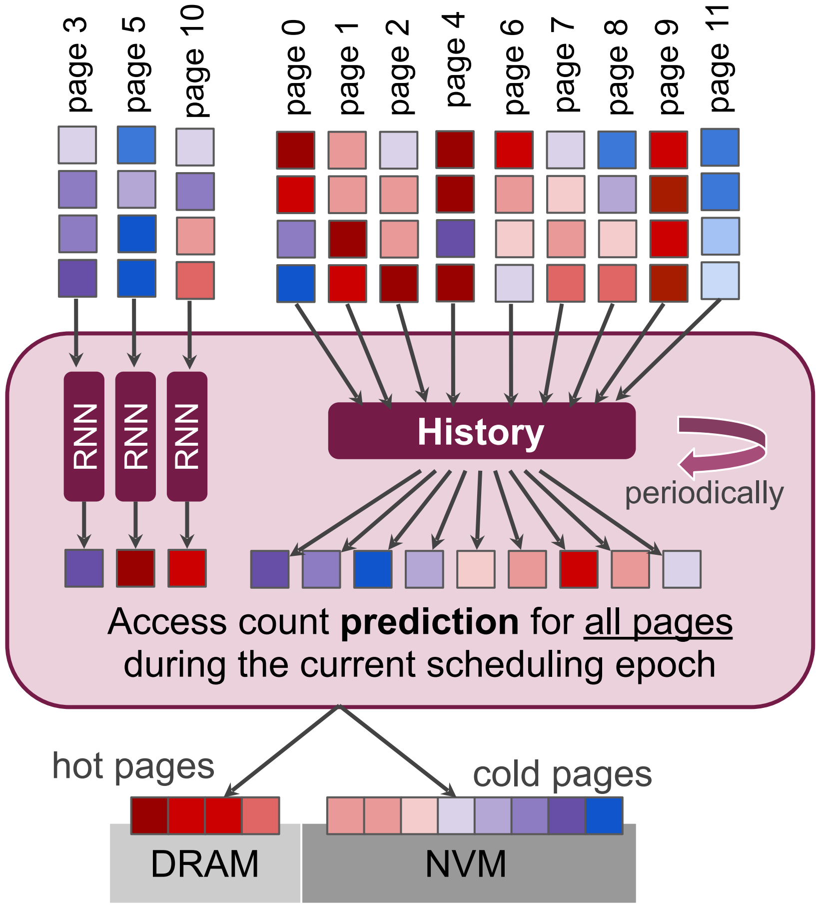
The Page Selector is run only **once**, to find the pages that require machine intelligence.

Step 2: RNN Training



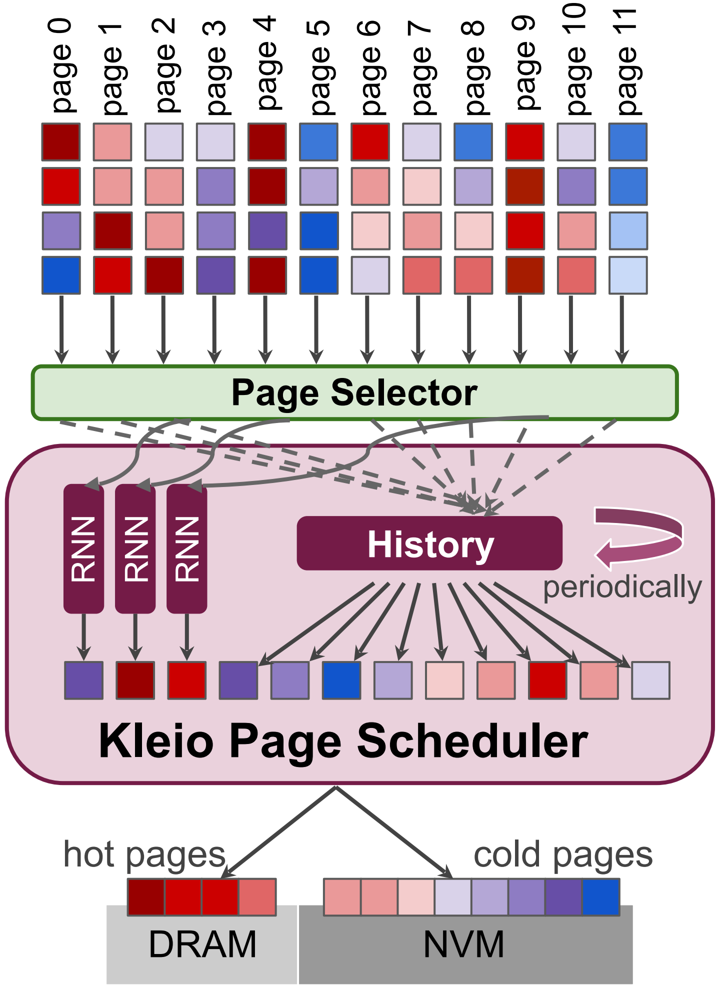
Trained models are saved.

Step 3: RNN Inference during Page Scheduling



Solution Overview

With some of the Implementation Details



Applications: CORAL, PARSEC, Rodinia
Number of pages: 8K - 800K
Number of Scheduling Epochs: up to 856 (x 1 sec)

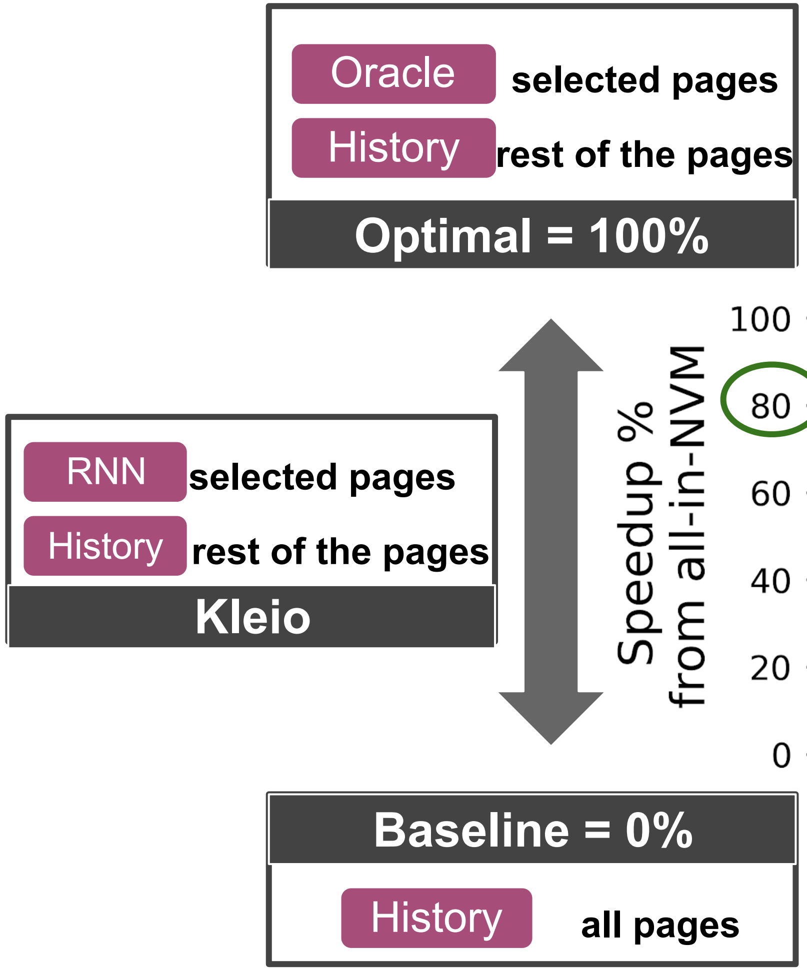
Memory Access Trace Collection:
IBS sampling and unsampled traces of Last Level Cache Misses
(time, virtual address, physical address, cpu core, thread id, load/store, hit/miss)

RNN Implementation:
Long Short Term Memory (LSTM) Networks, Keras API, Tensorflow Backend
(more on the paper!)

Hybrid Memory System:
Trace-based analysis for DRAM hit rates.
Analytical model to extrapolate runtime based on access distribution across DRAM and NVM assuming zero cost migrations.

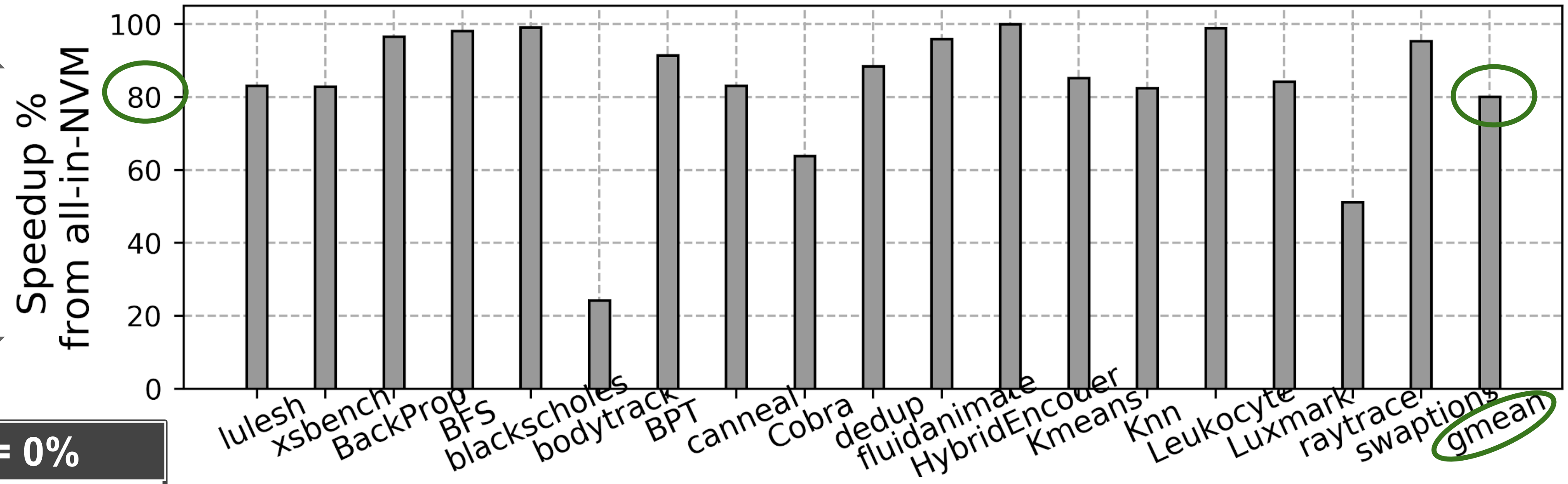
Evaluation

Kleio closes on average 80% of the performance gap



The higher
The better

More than **95%** for **half** of the applications!



For fixed DRAM:NVM capacity.
For 100 selected pages.

Evaluation
Practical Considerations

Resource Utilization **per RNN model** on general purpose CPU:

Training ⏰ 2 hours 💾 Tens GBs of Memory | **Inference** ⏰ 3-4 sec 💾 0.5 MB of Storage

✅ **Duration** can be further reduced by multiple orders of magnitude with anticipated ML accelerators.

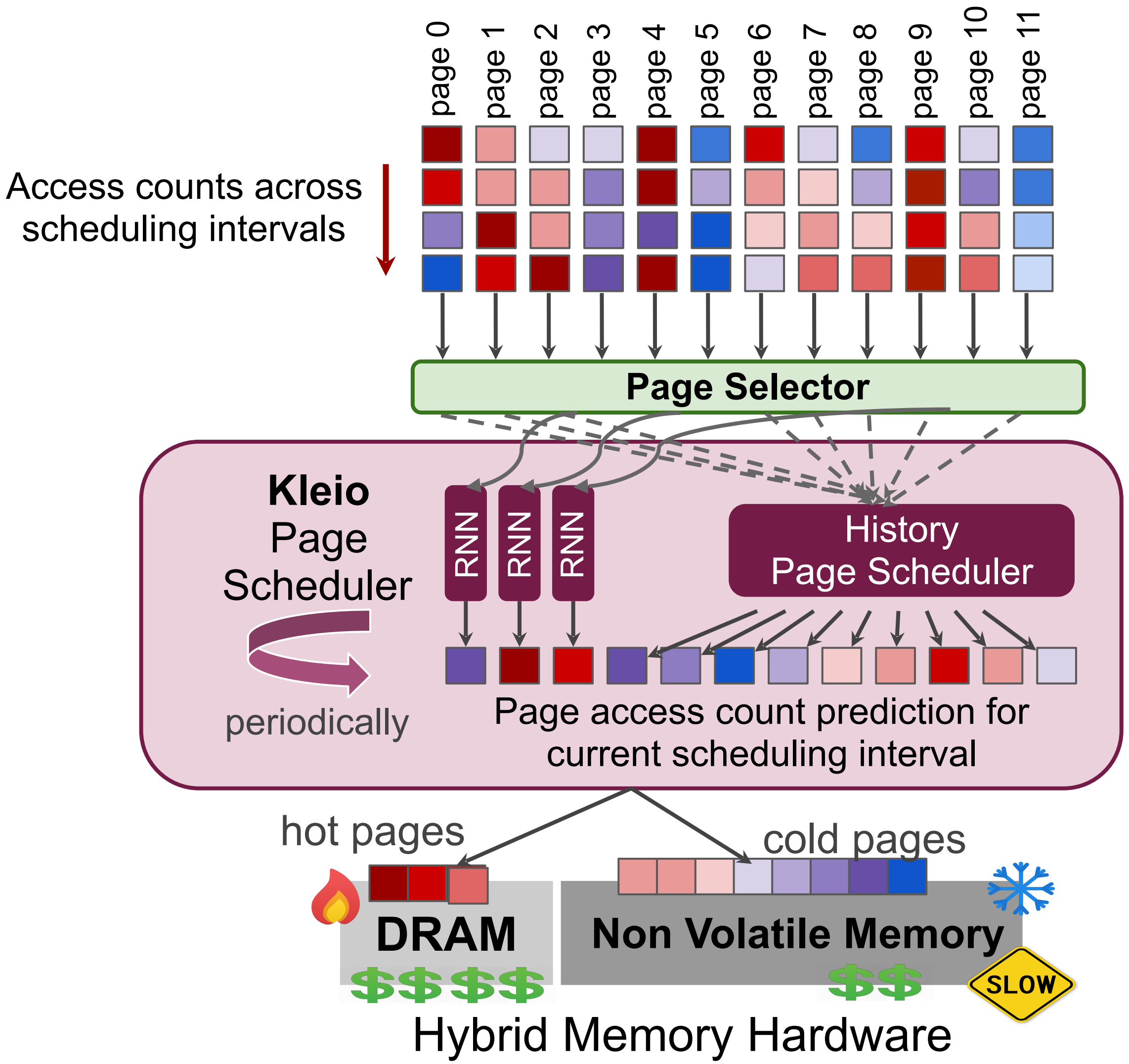
✅ Large **memory** footprint can be accommodated by hybrid memory systems!

✅ Kleio's Page Selector already drastically reduces the **problem space**.

✅ RNNs can also be trained in an **online** manner.

➡️ **There is great potential for Kleio to be adapted in an online practical system-level solution.**

Summary
Paper Contributions

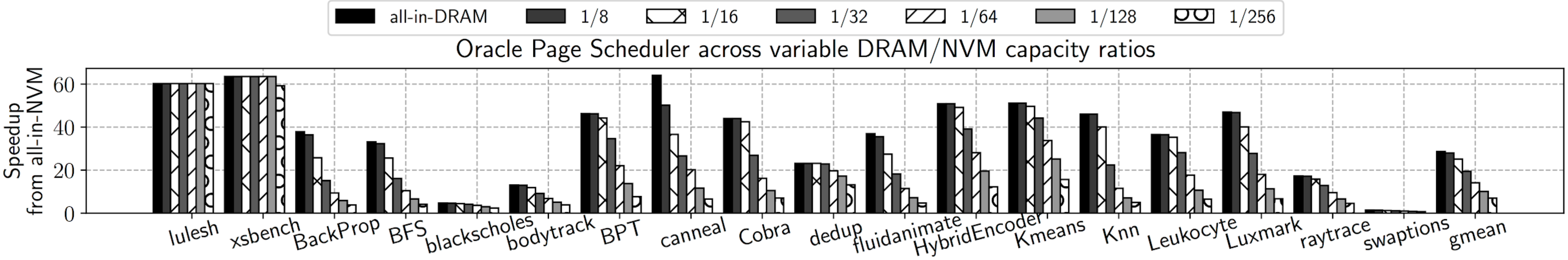


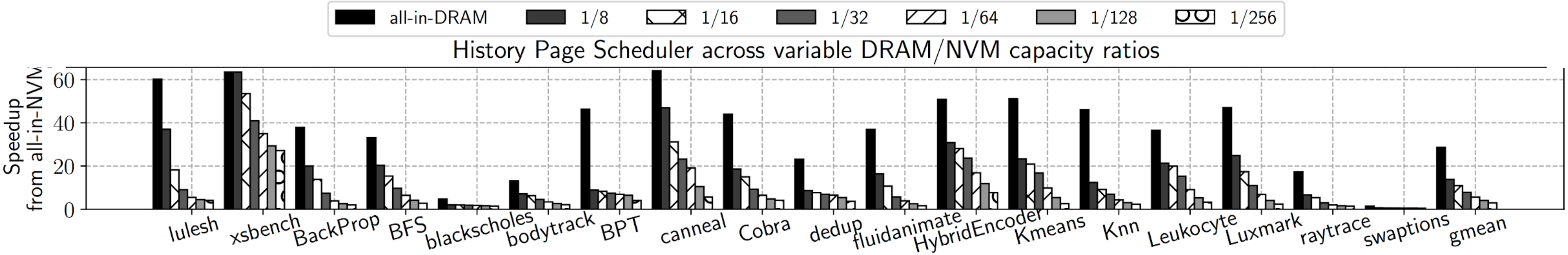
***Kleio** is a machine intelligent page scheduler for hybrid memory systems.*

- ✅ Bridges the existing **performance gap by 80%**.
- ✅ Cleverly identifies the **page subset** whose timely allocation in DRAM will boost performance via machine intelligent placement.
- ✅ Lays the ground for **practical integration** of machine intelligent memory management solutions in future systems.

Backup Slides

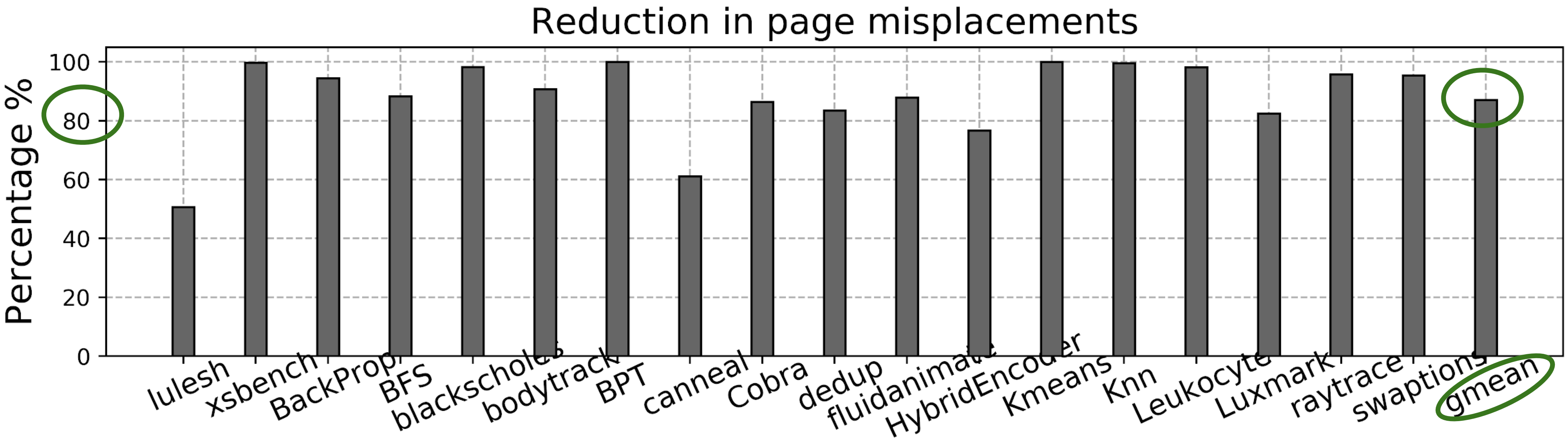
Performance Gap





Evaluation

Kleio reduces on average 85% of the page misplacements

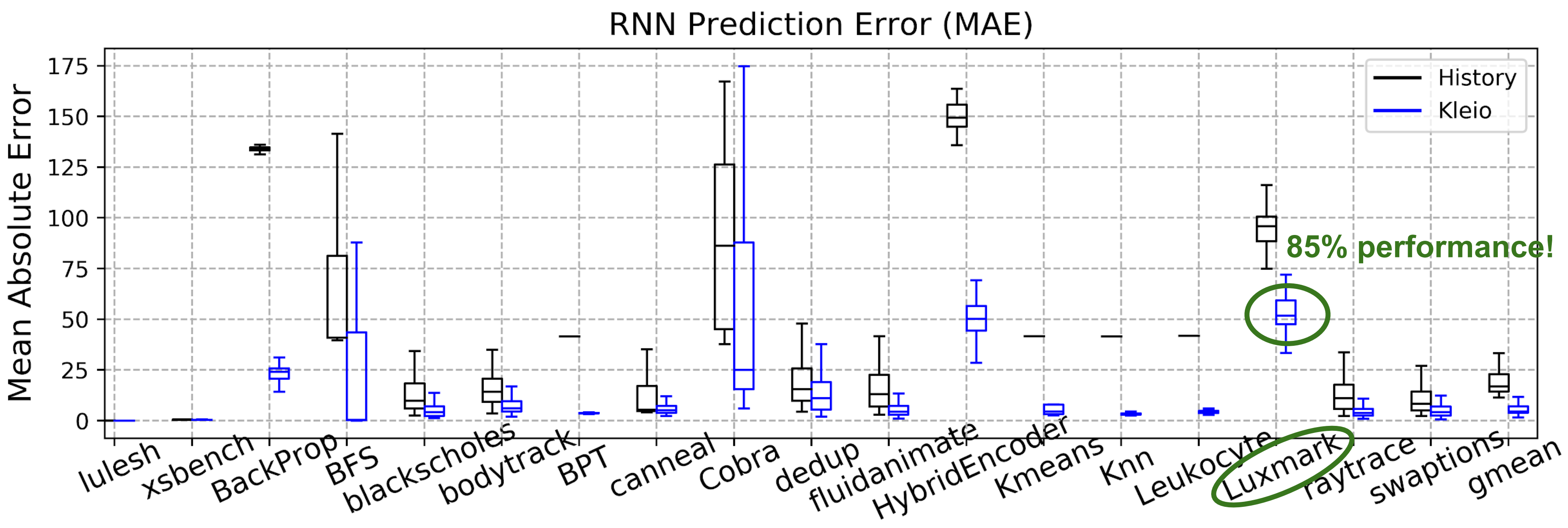


The higher
The better

For the pages managed with Machine Intelligence.

Evaluation

RNN Prediction Accuracy does not impact application performance directly



The lower
The better

e.g. MAE = 50 means that the RNN predicted on average 50 more accesses per scheduling epoch per page.

Important: High MAE doe not impact performance, **when it does not affect the placement decision**.

✅ Kleio is robust against RNN mispredictions.

Evaluation
Comparison with Other Solutions

Kleio’s deployment requirements:

Memory access trace collection + RNN training. | RNN inference part of page scheduling decision making.

Existing Solutions:

- Offline profile solutions: *X-Mem* [Eurosys ‘16] *Dataplacer* [ISMM ‘16]

  - Provide only static placement.

- Dynamic solutions: *Unimem* [SC ‘17] (MPI phase profiling) *Tahoe* [SC ‘ 18] (Task profiling)

  - Rely on application phase-changing behavior and detection.

✅ **Kleio works for any application and provides dynamic data management.**

DISCLAIMER & ATTRIBUTION

The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

ATTRIBUTION