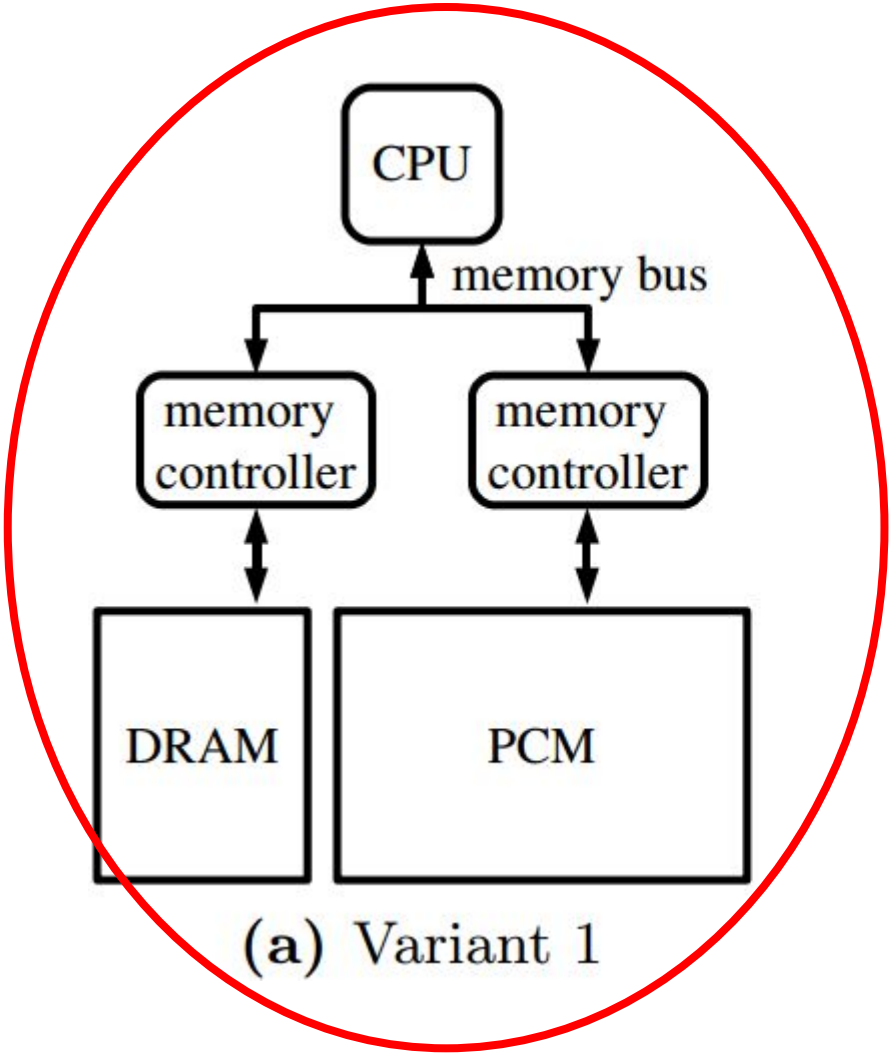
Wear-Aware Algorithms for PCM-Based Database Buffer Pools

(Revisited)

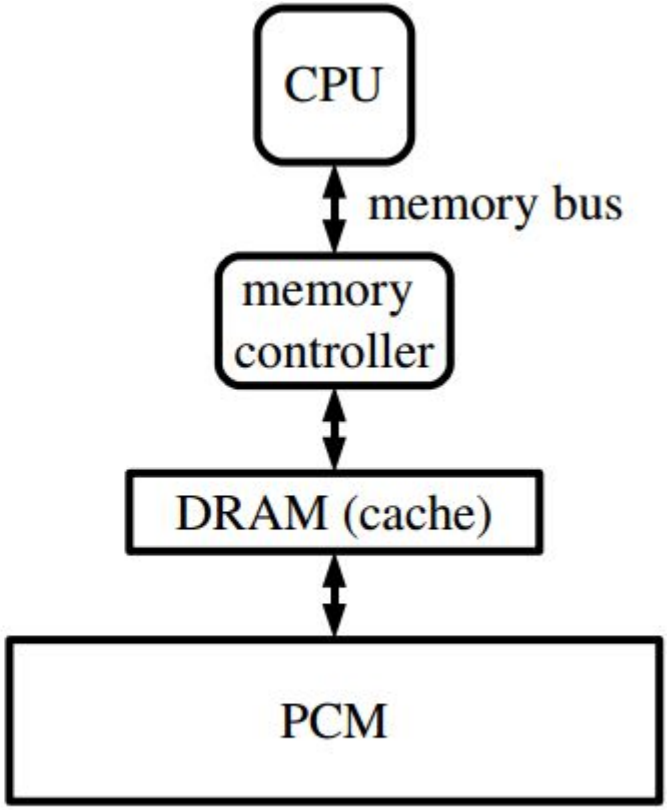
PCM

- Phase Change Memory
- Non-volatile
- Bit alterable & Byte addressable
- Ideal to be used in main memory system

Hierarchy Architecture



(a) Variant 1



(b) Variant 2

Problem

- Write Endurance
  - PCM > Flash Memory
- But…
  - buffer pool is expected to have high write traffic
- Goal
  - wear-leveling
  - prevent high-traffic writes to same PCM location

Yi Ou's Work

- *wear unit: u*
  - 4 Bytes

- writes in a page are rarely uniformly distributed
  - parts of a page will worn out sooner than other parts

- *wear count: w(u)*
- *page wear: ∑w(u)*

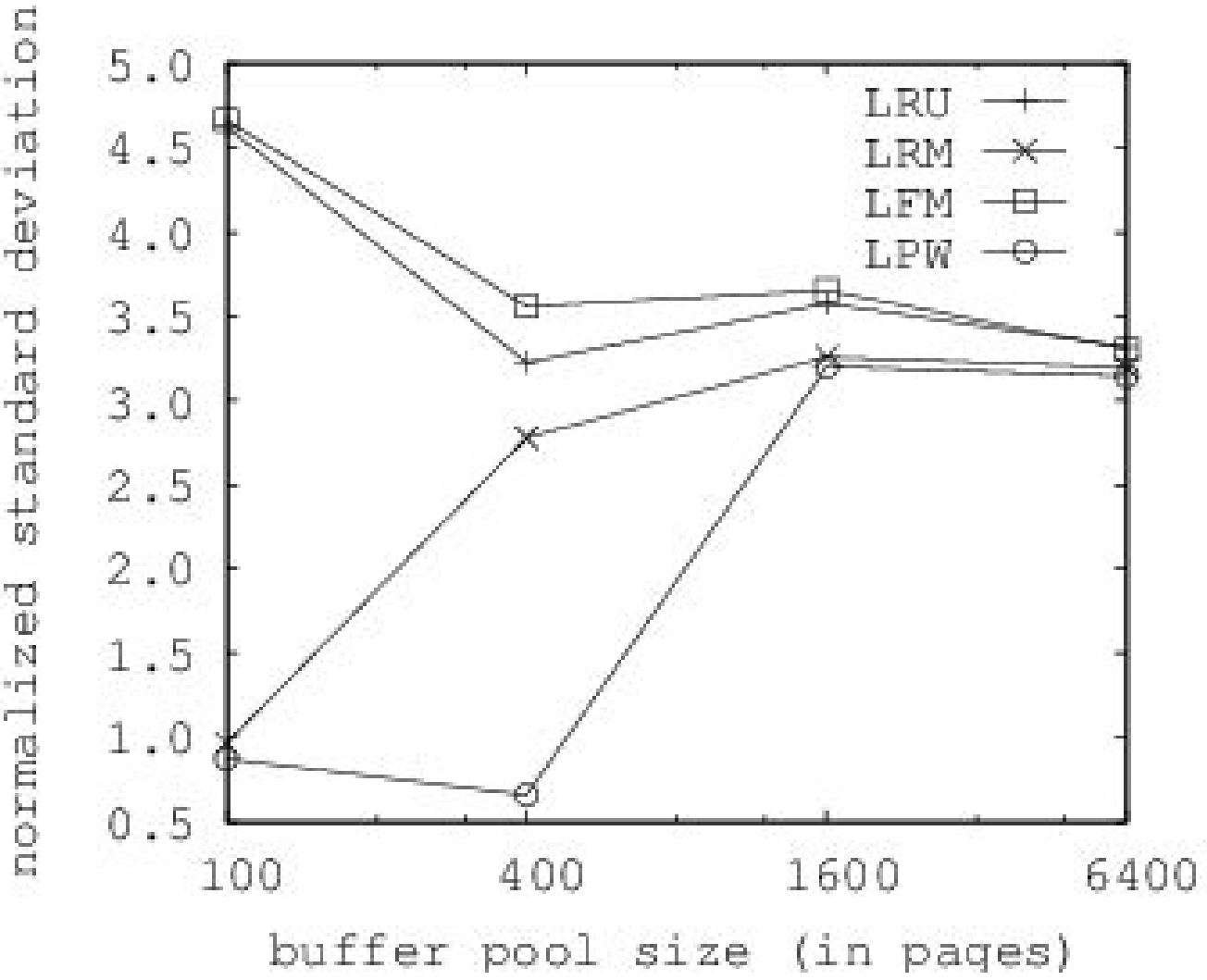
Yi Ou's Work

- LPW: Least Page Wear
  - pick victim with smallest page wear
- LFM: Least Frequently Modified
  - LFU, but do not reset frequency counter at page replacement
- LRM: Least Recently Modified
  - LRU, but page hits do not change position in list

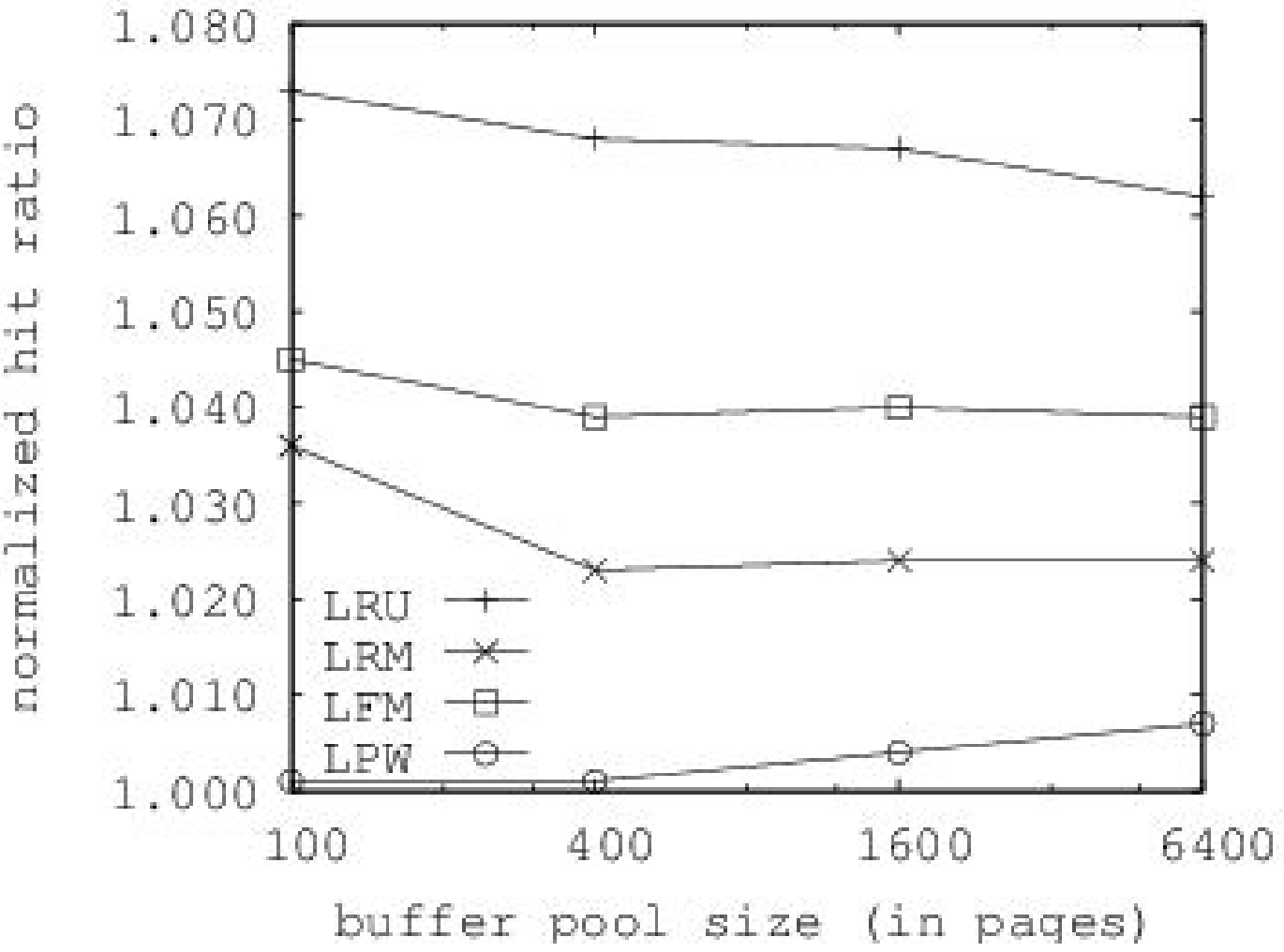
Yi Ou's Experiments

- Wear-leveling efficiency
  - standard deviation of wear count
- Hit Ratio
- Environment
  - 1GB database
  - 8KB page size
  - 4B wear unit
- Workload
  - 1M record requests
  - 80% reads, 20% updates
  - Skewed access to pages and records: 80 20

Yi Ou's Results (Standard Deviation)



Yi Ou's Results (Hit Ratio)



Yi Ou's Conclusions

- LPW best for wear-leveling, but poor hit ratio
- LRU best hit-ratio, but poor wear-leveling
- LRM as a middle-ground
- Conflict between wear-leveling and hit ratio

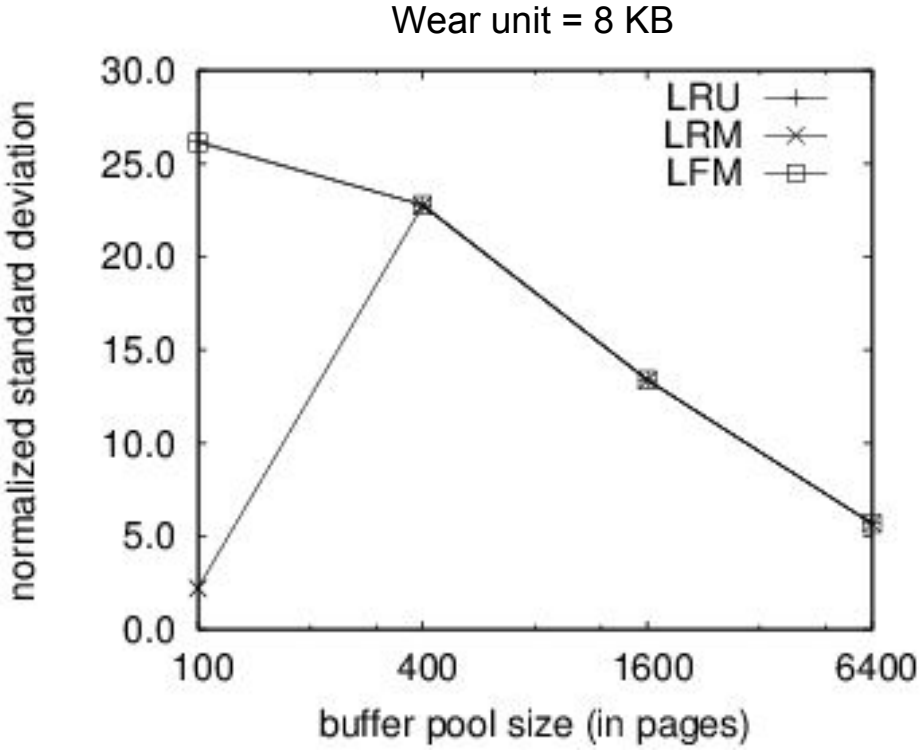
Yi Ou, Problem #1

- Wear unit counter: integer (4 Bytes)
  - 100% overhead

- w.r.t. distribution of writes inside a page
  - the writes happen to frames in the buffer pool
  - it is unlikely that all pages loaded into the same frame will have the exact same portion written more frequently

Problem #1, Solution #1

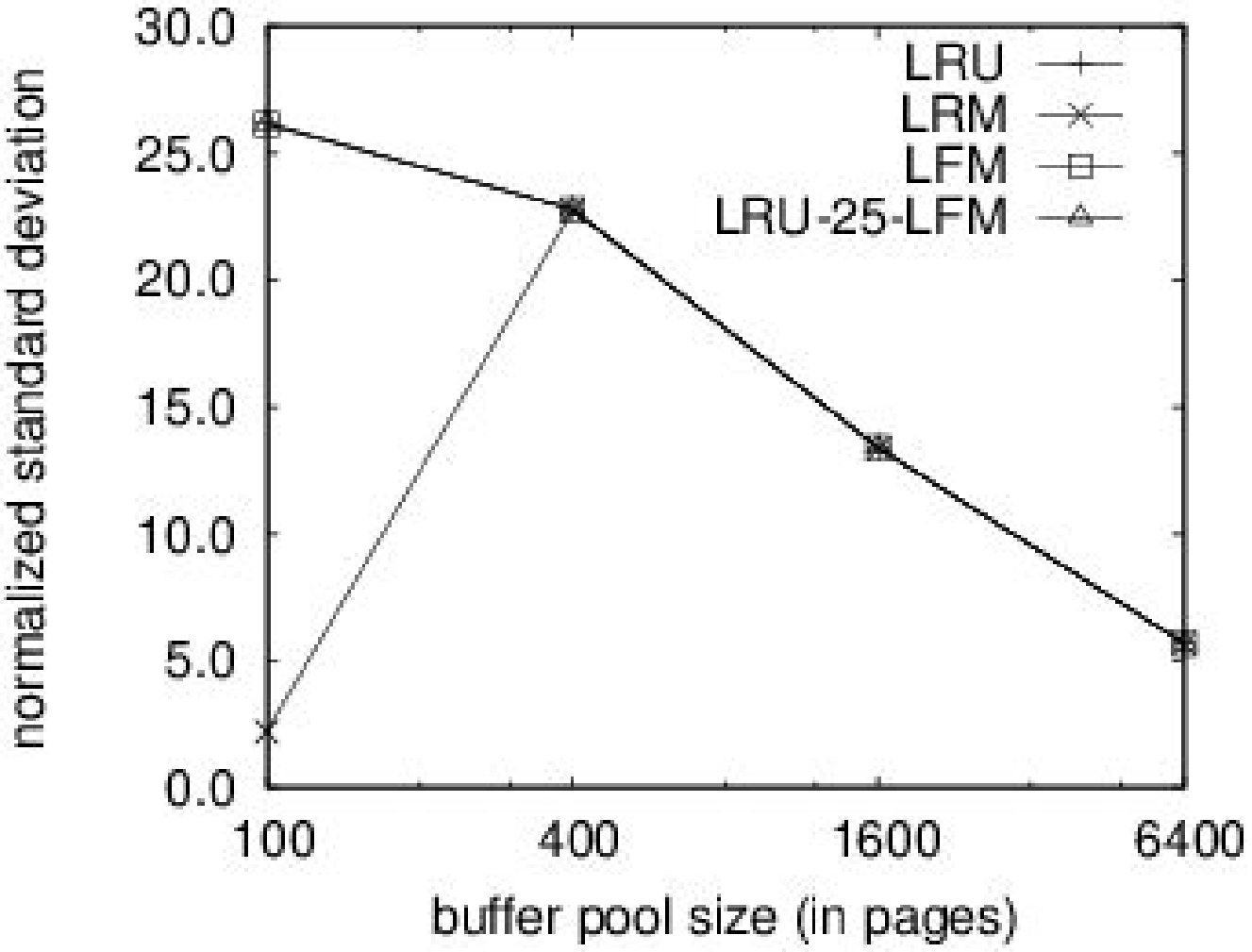
- Wear-unit size == page size
  - LPW == LFM



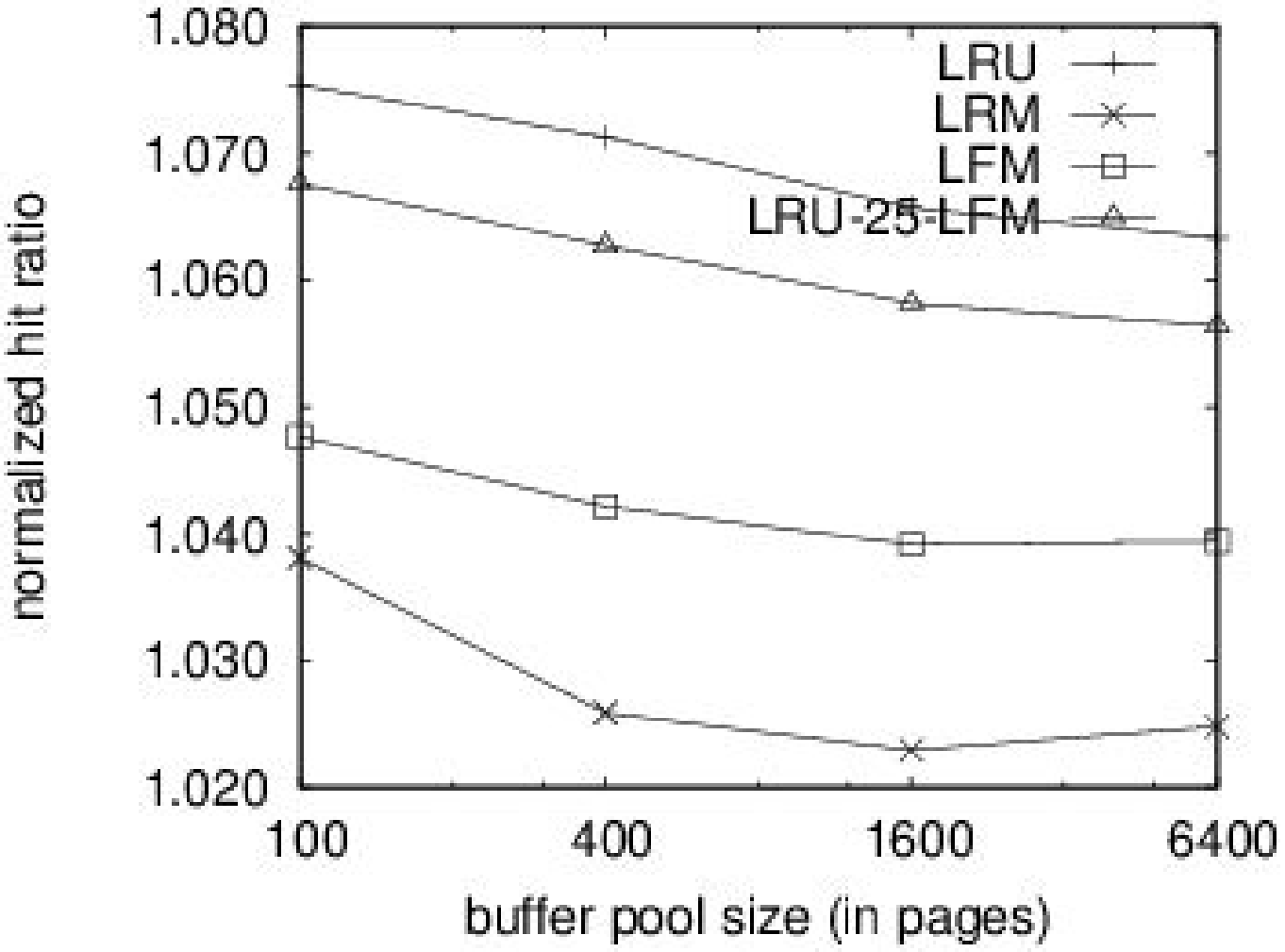
Problem #1, Solution #2

- Hybrid algorithm
  - LRU-W-LFM
  - For W=25:

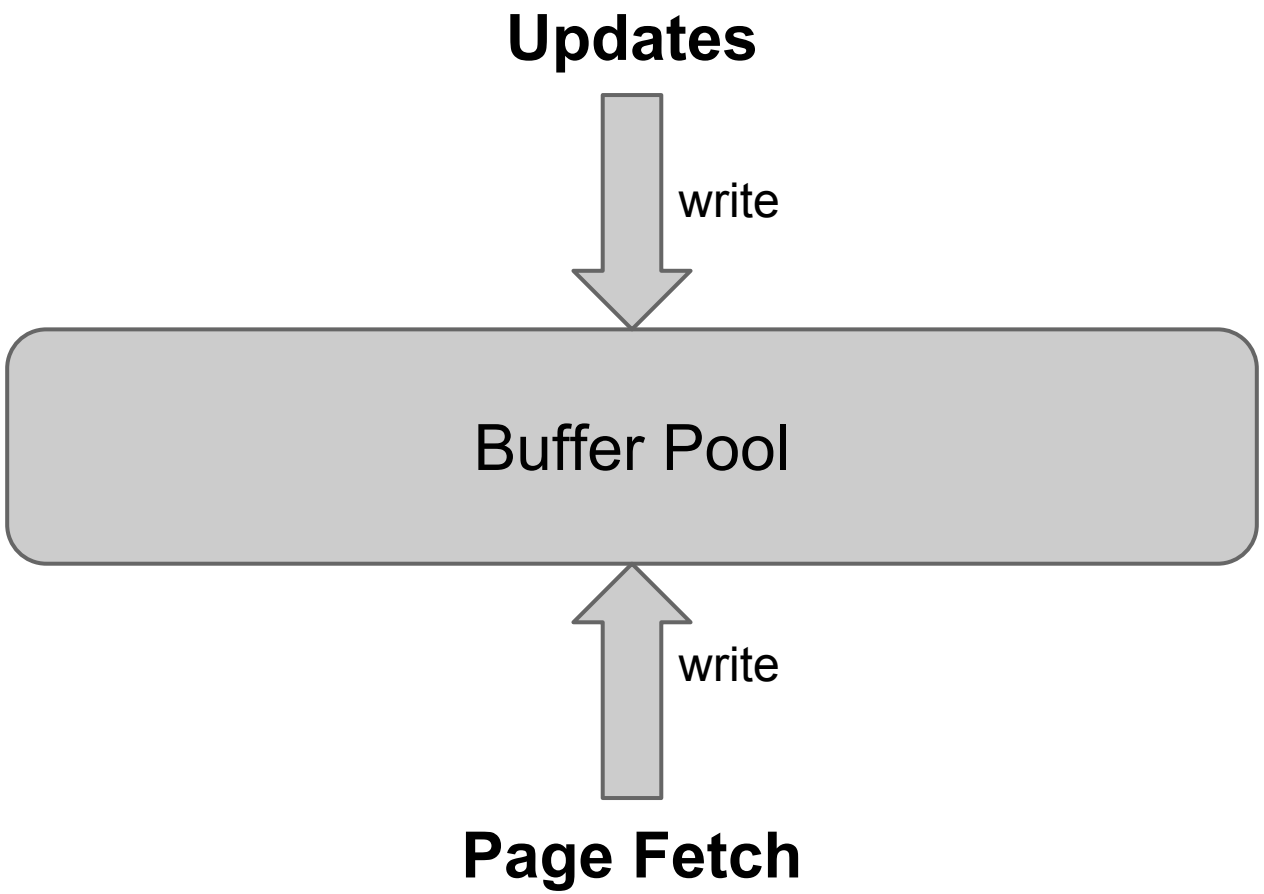
Problem #1, Solution #2



Problem #1, Solution #2



Yi Ou, Problem #2



Yi Ou, Problem #2

- Hot pages
  - always in the buffer
  - very often updated
  - never replaced

- Our conclusion:
  - replacement algorithm is not the place to implement wear-leveling
  - SWAP-strategy

SWAP

- Yi Ou:
  - “[...] wear leveling and hit ratio are two conflicting goals [...]”

- Decouple wear-leveling control from replacement algorithm

SWAP

- Keep track of #writes of each buffer frame

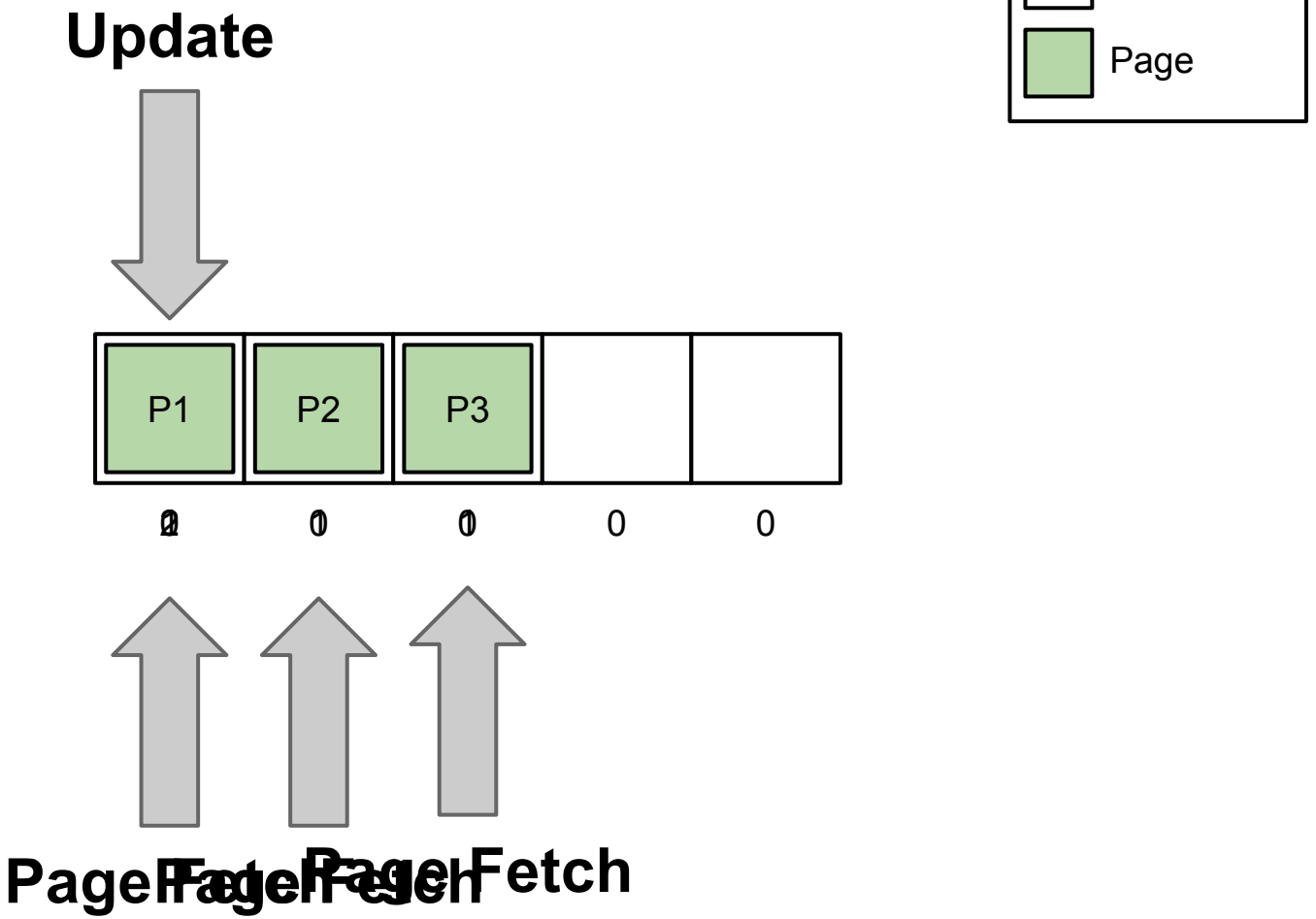
- Whenever a write happens:
  - If #write of the frame is a certain threshold above the average of writes:
    - SWAP the frame content with another frame

SWAP

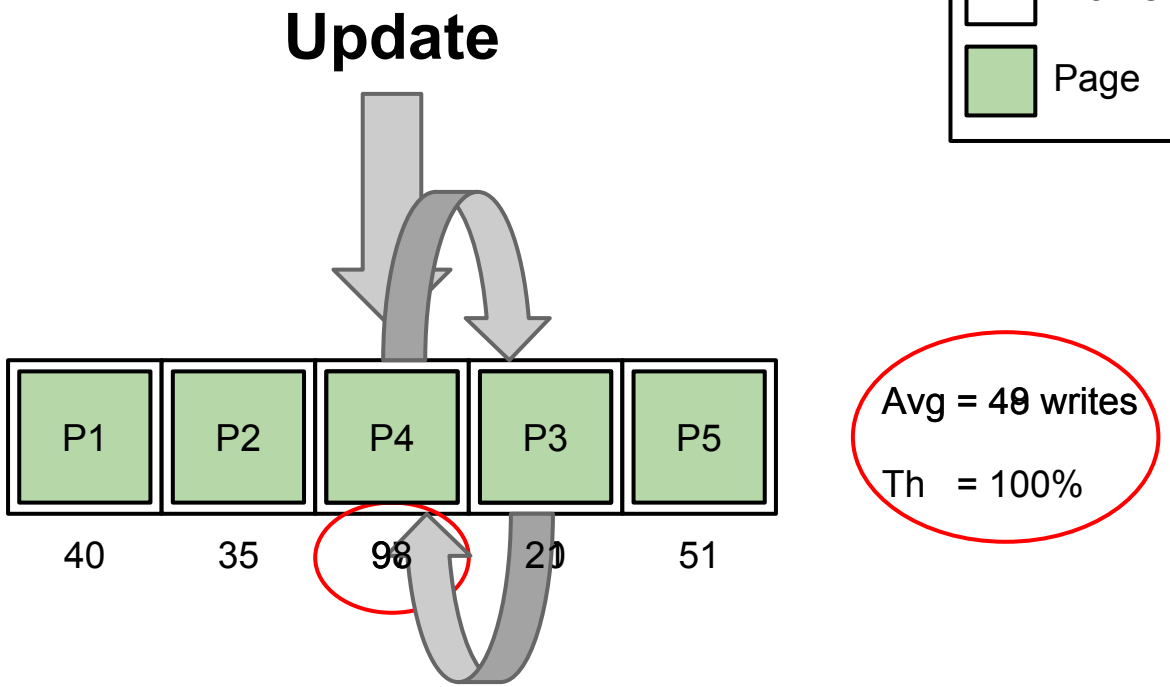
- SWAP with whom?
  - Clock-like structure
  - First frame found that has the #writes <= average

- What is a good threshold?
  - 100% above the average of writes

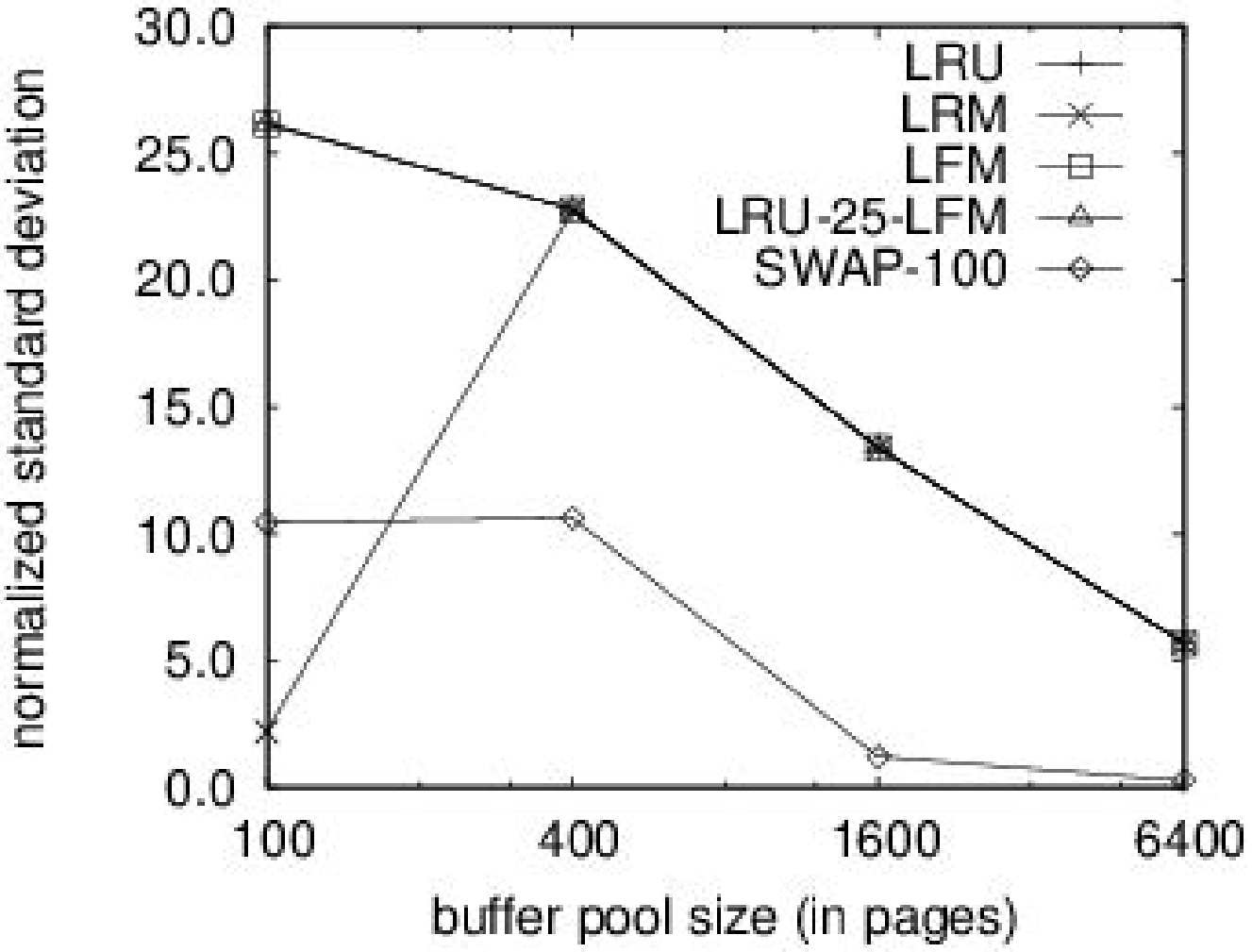
SWAP Example



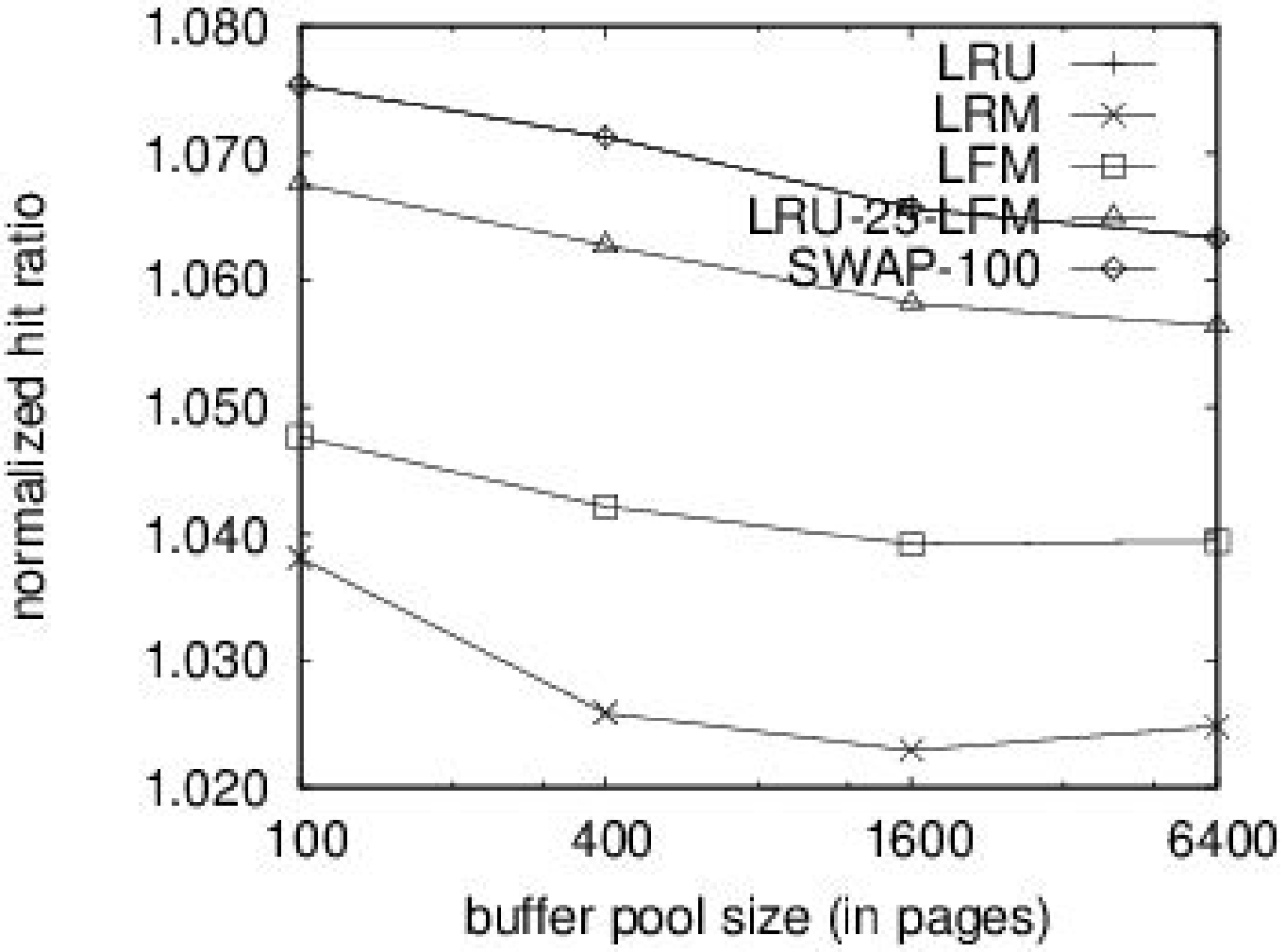
SWAP Example



SWAP Results (Standard Deviation)



SWAP Results (Hit Ratio)



Conclusion

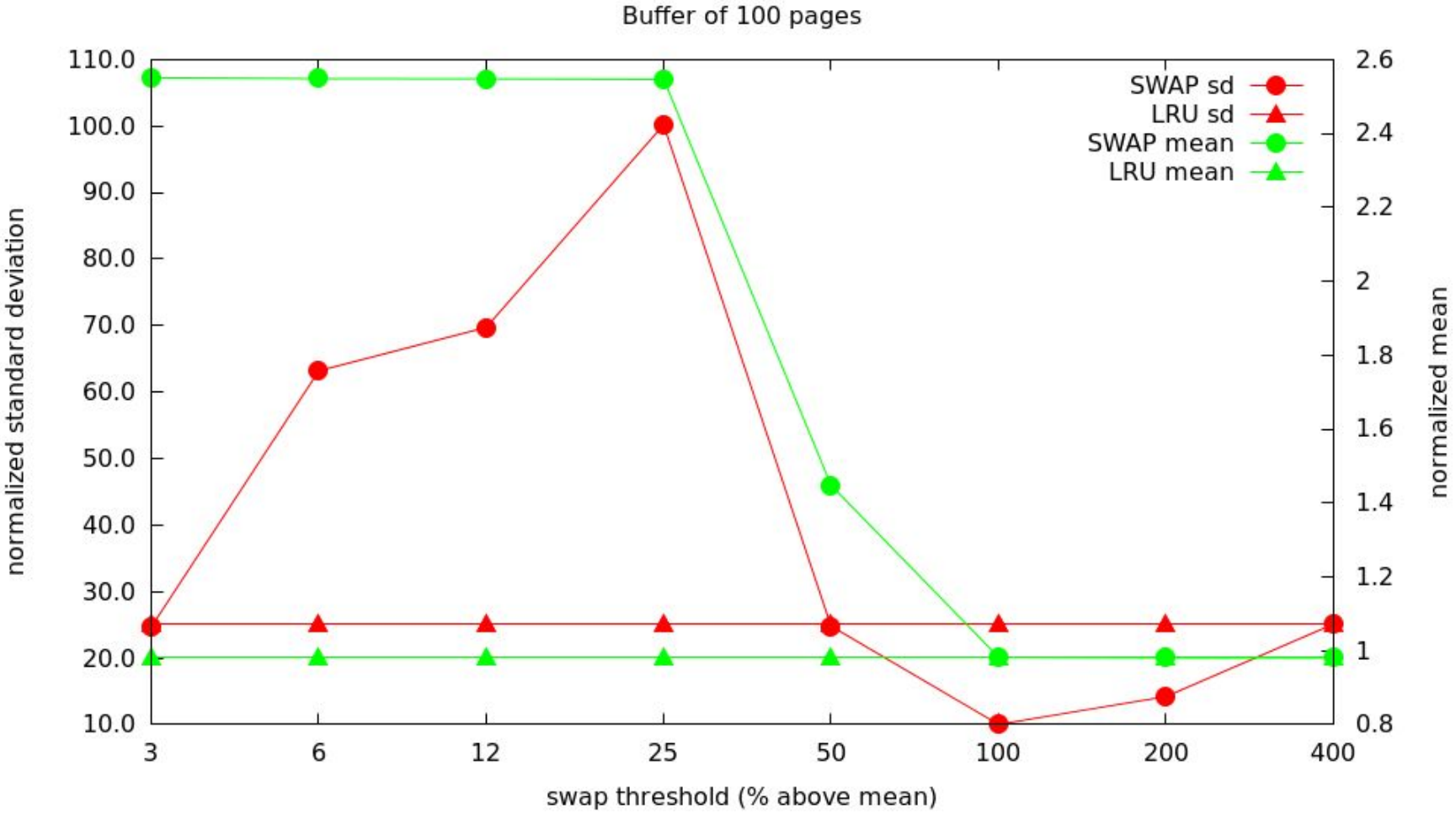
- Wear-unit size = Page size
- SWAP is very simple and provides
  - Same hit ratio as LRU
  - The best distribution of writes
- Wear leveling != page replacement

The End

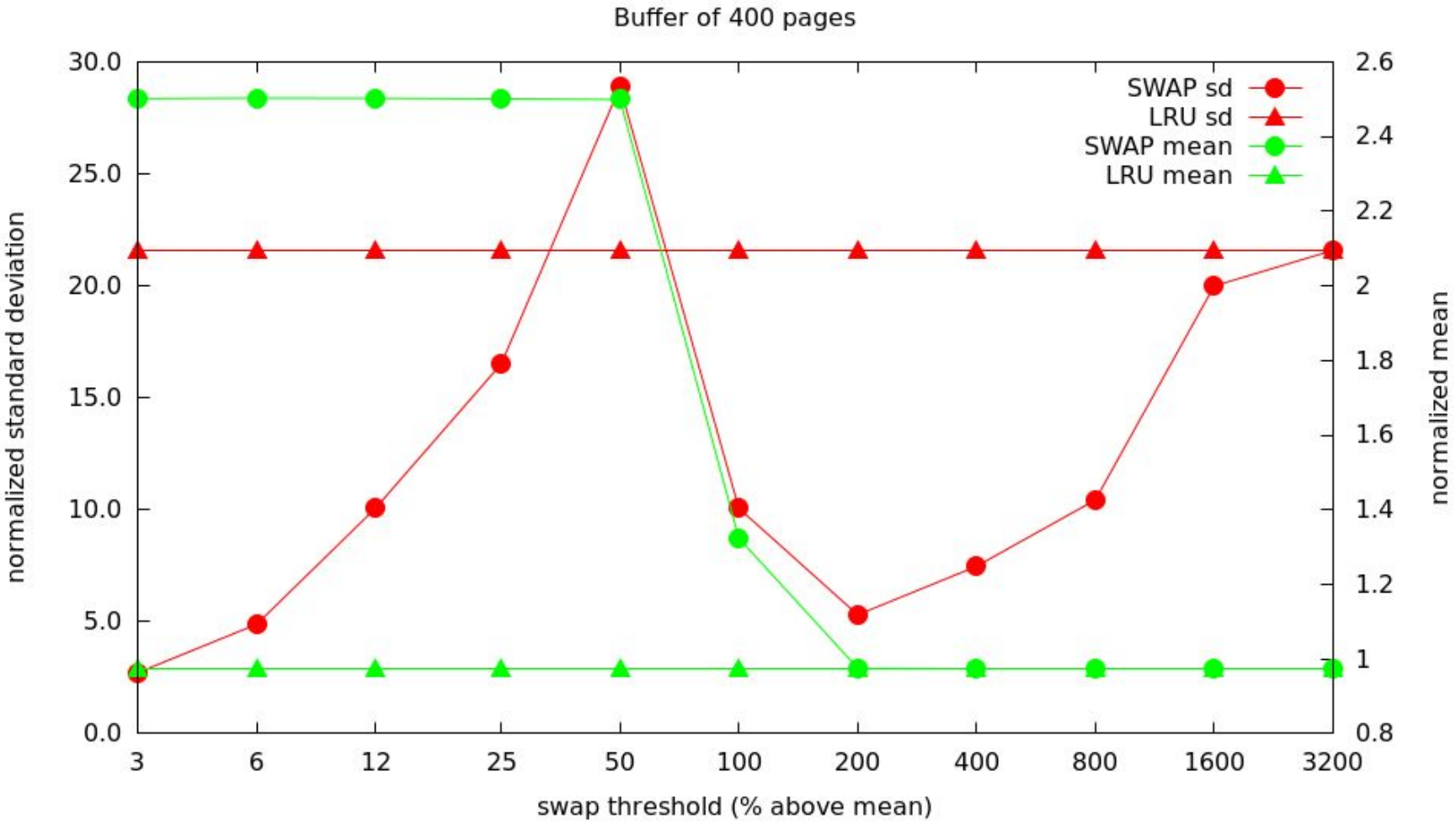
- Source code available at:

  https://bitbucket.org/lslersch/eessd

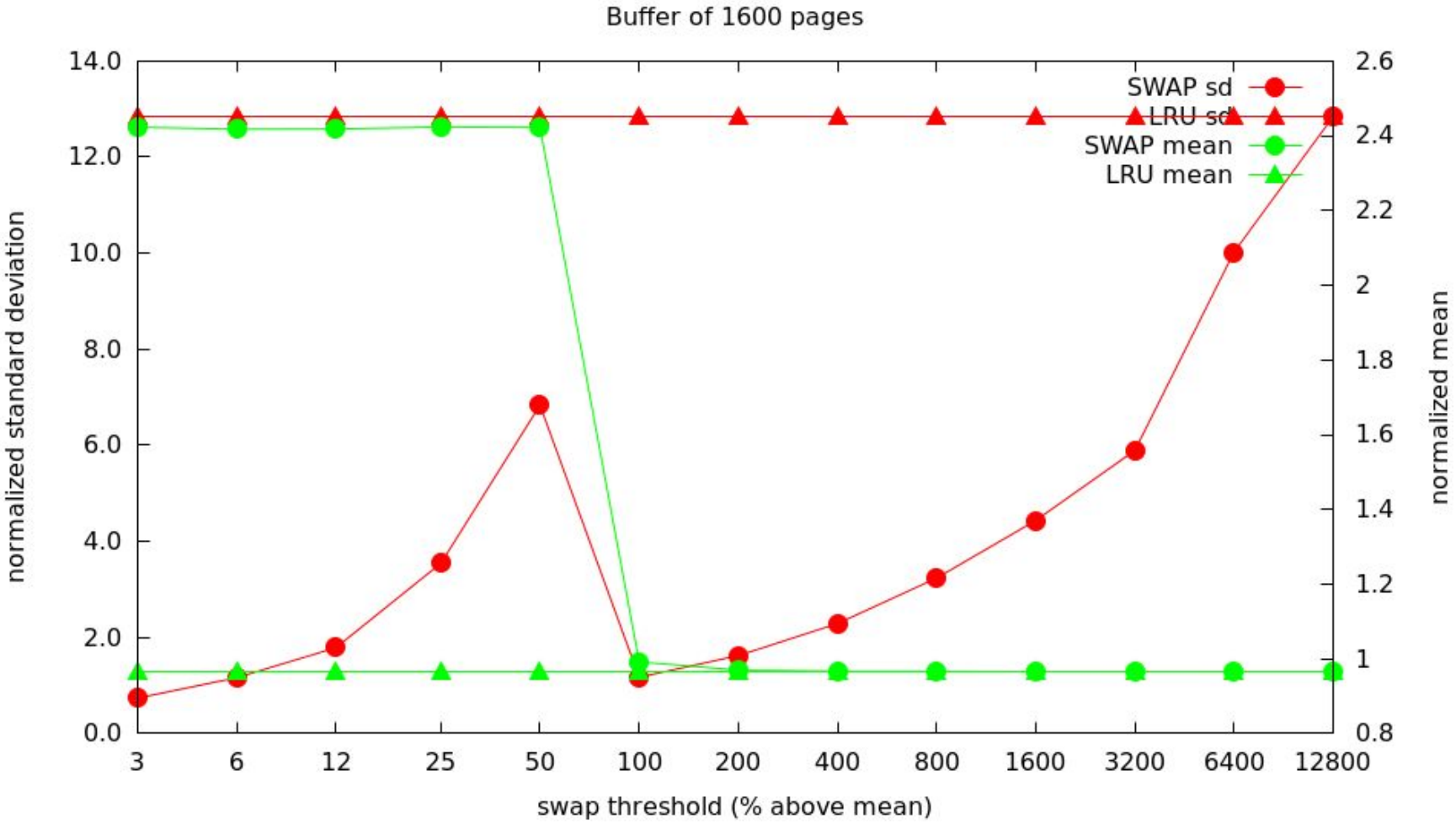
Determining Threshold



Determining Threshold



Determining Threshold



Determining Threshold