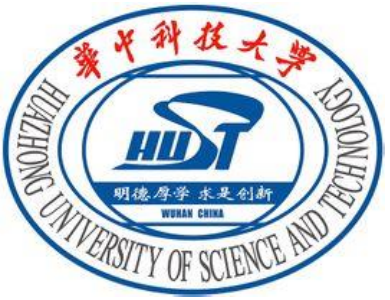


An Efficient Wear-level Architecture using Self-adaptive Wear Leveling

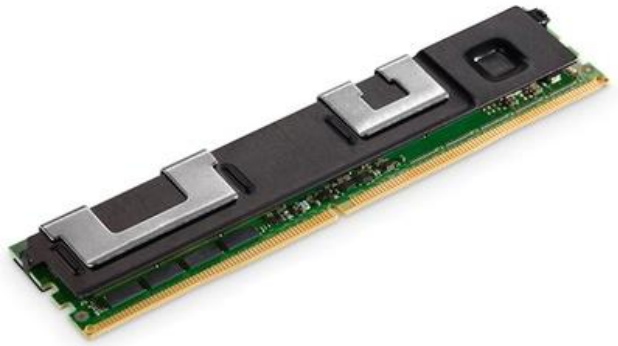
Jianming Huang, Yu Hua, Pengfei Zuo, Wen Zhou, Fangting Huang

Huazhong University of Science and Technology

ICPP 2020

Non-volatile Memory

- NVM features
  - Non-volatility
  - Large capacity
  - Byte-addressability
  - DRAM-scale latency

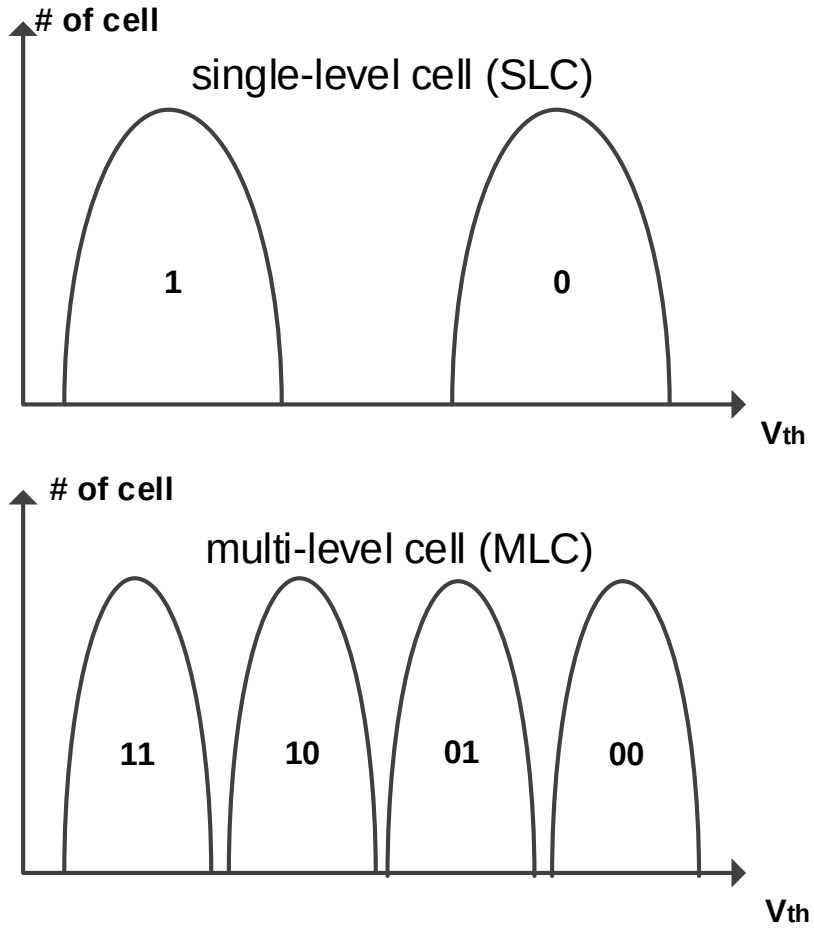


Intel Optane DC Persistent Memory

- NVM drawbacks
  - Limited endurance
  - High write energy consumption

Multi-level Cell NVM

The MLC technique has been used in different kinds of NVM, including PCM, RRAM, and STT-RAM.



Wear-leveling schemes are necessary and important

Compared with SLC NVM, MLC NVM

- Higher storage density
- Lower cost
- Comparable read latency
- Weaker endurance

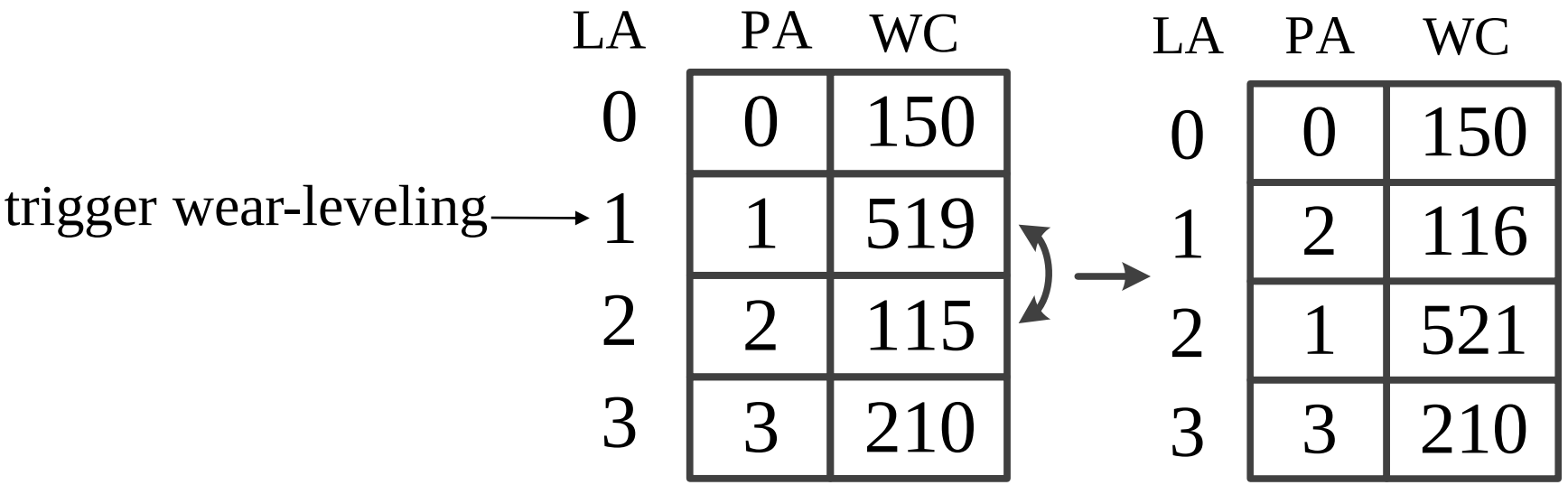
10^7 of SLC PCM vs 10^5 of MLC PCM

Wear-leveling Schemes

- Table based wear-leveling scheme (TBWL)
- Algebraic based wear-leveling scheme (AWL)
- Hybrid wear-leveling scheme (HWL)

Wear-leveling Schemes

- Table based wear-leveling scheme (TBWL)

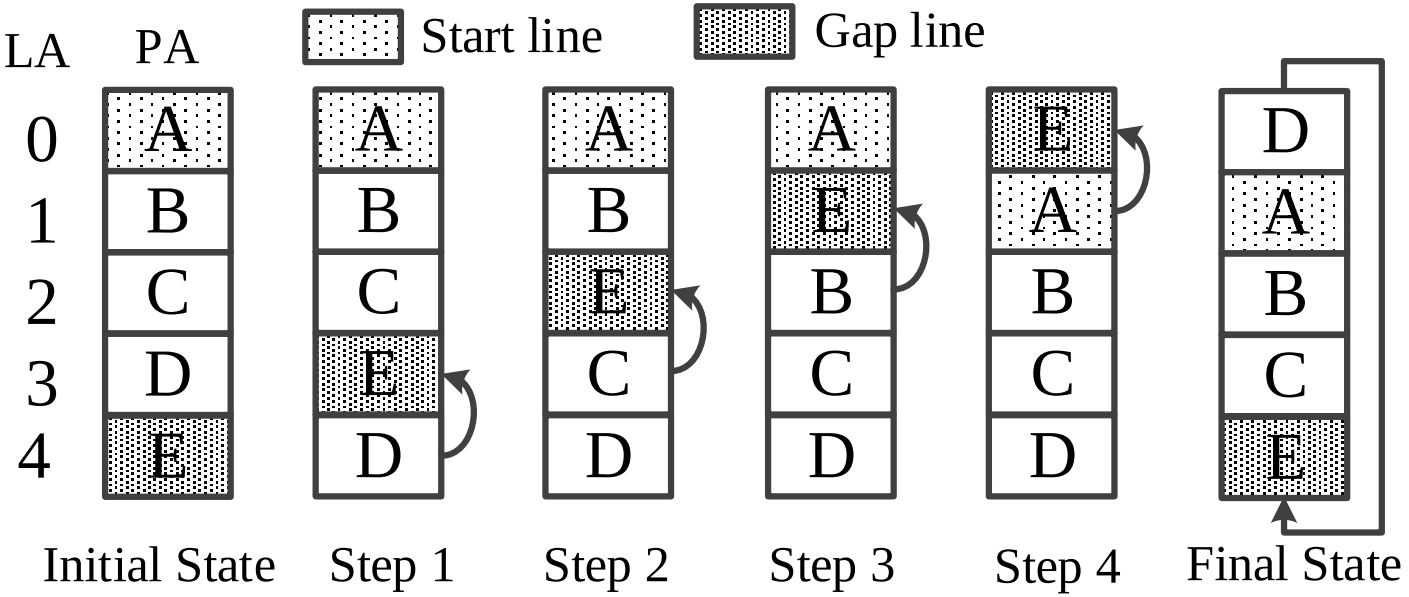


Segment Swapping

Wear-leveling Schemes

- Table based wear- leveling scheme (TBWL)

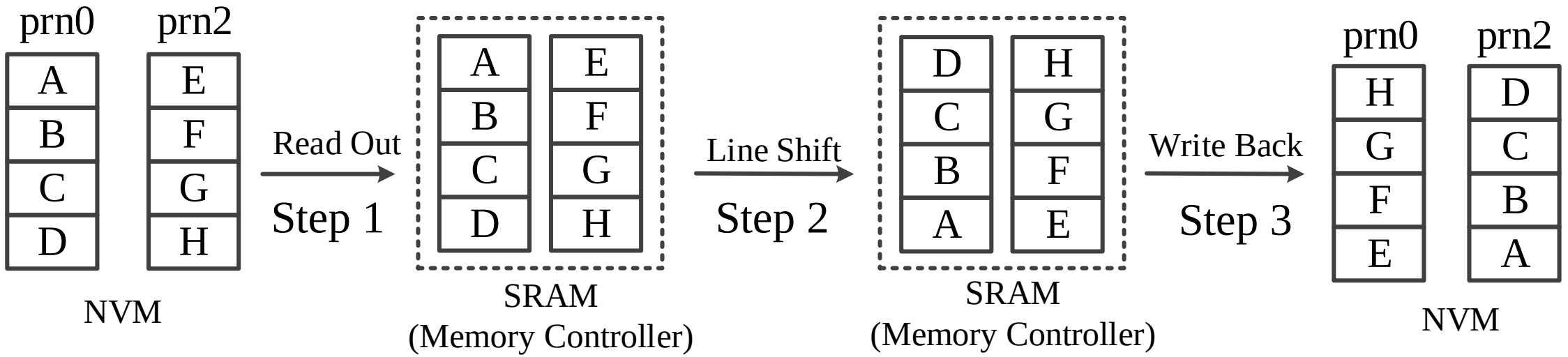
- Algebraic based wear-leveling scheme (AWL)



region-based Start-Gap (RBSG)

Wear-leveling Schemes

- Table based wear- leveling scheme (TBWL)
- Algebraic wear-leveling scheme (AWL)
- Hybrid wear-leveling scheme (HWL)



PCM-S

RAA Attack for TBWL

➢ RAA attack

- Repeated Address Attack (RAA)

- Repeatedly write data to the same address

➢ The lifetime of TBWL under RAA attack

- One region contains the limited lines

- All lines in one region are repeatedly written

- NVM is worn out at the early stage
  - $(The$ number of lines within a region$) \times ($The endurance of a line$)$

BPA Attack for AWL

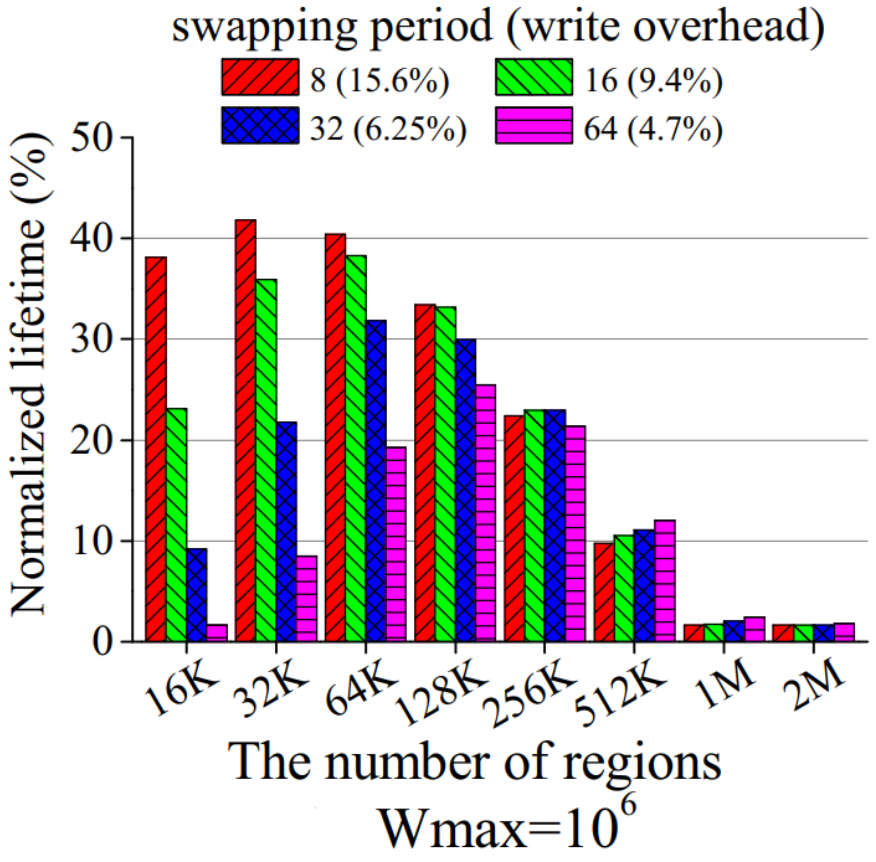
➢ BPA attack

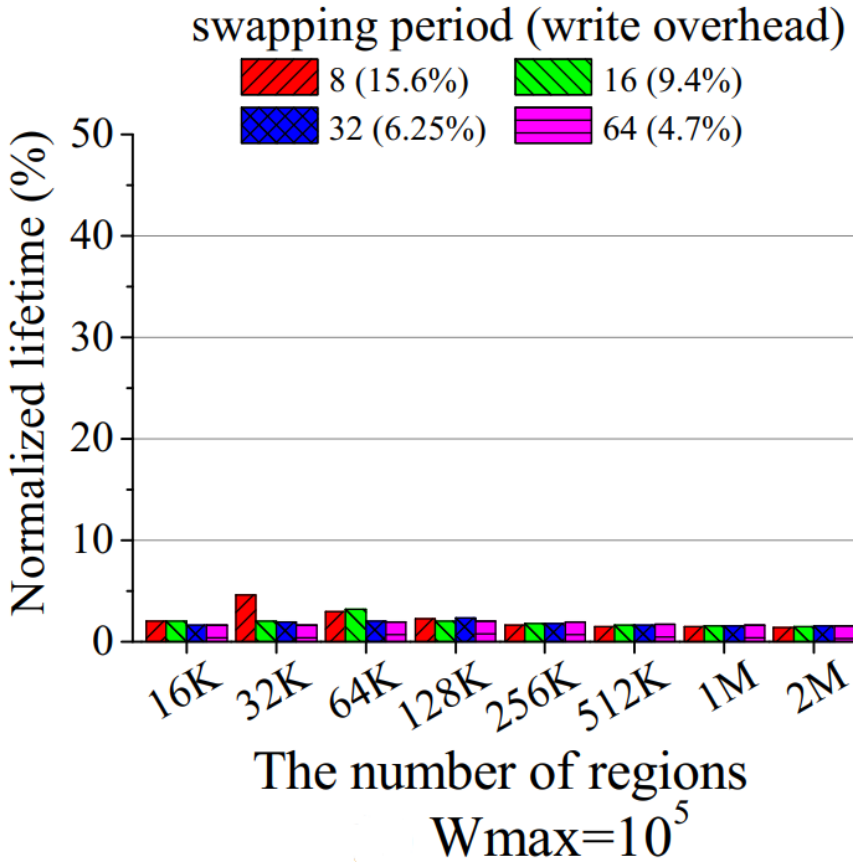
- Birthday Paradox Attack (BPA)

- Randomly select logical addresses and repeatedly write to each

BPA Attack for AWL

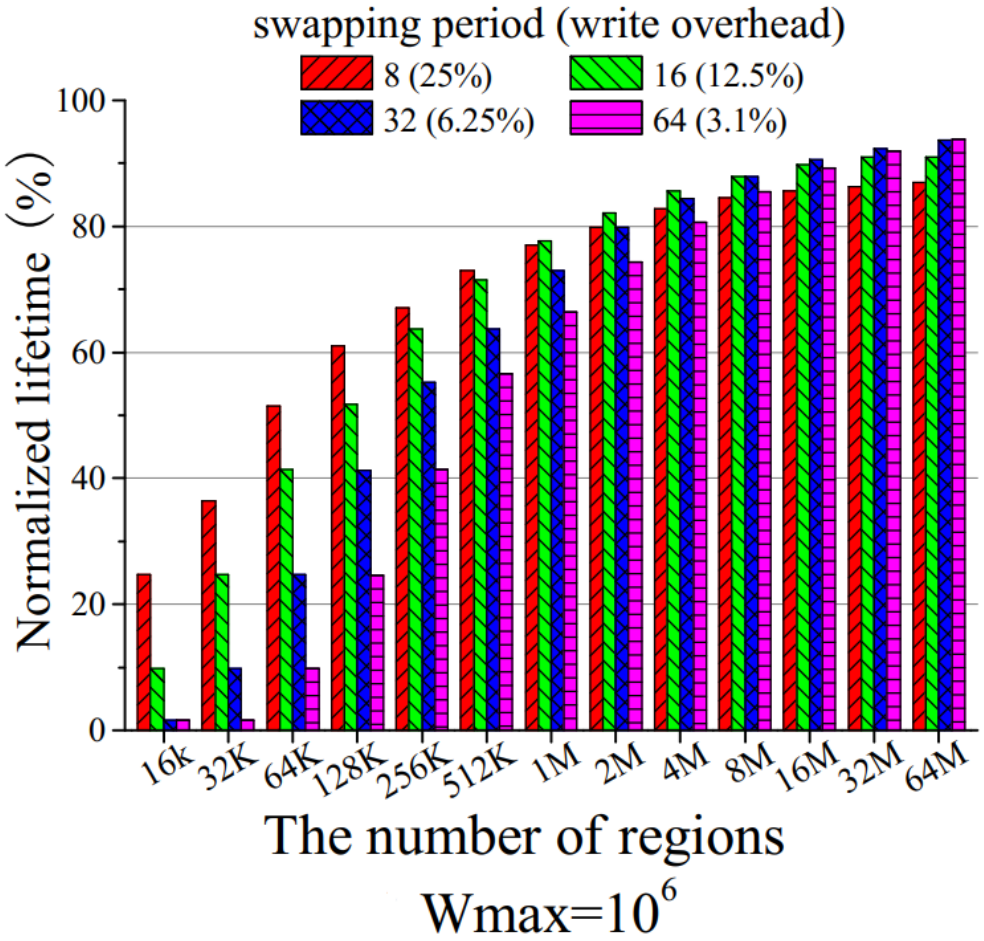
- **BPA attack**
- **The lifetime of AWL under BPA attack**
  - Lifetime is low

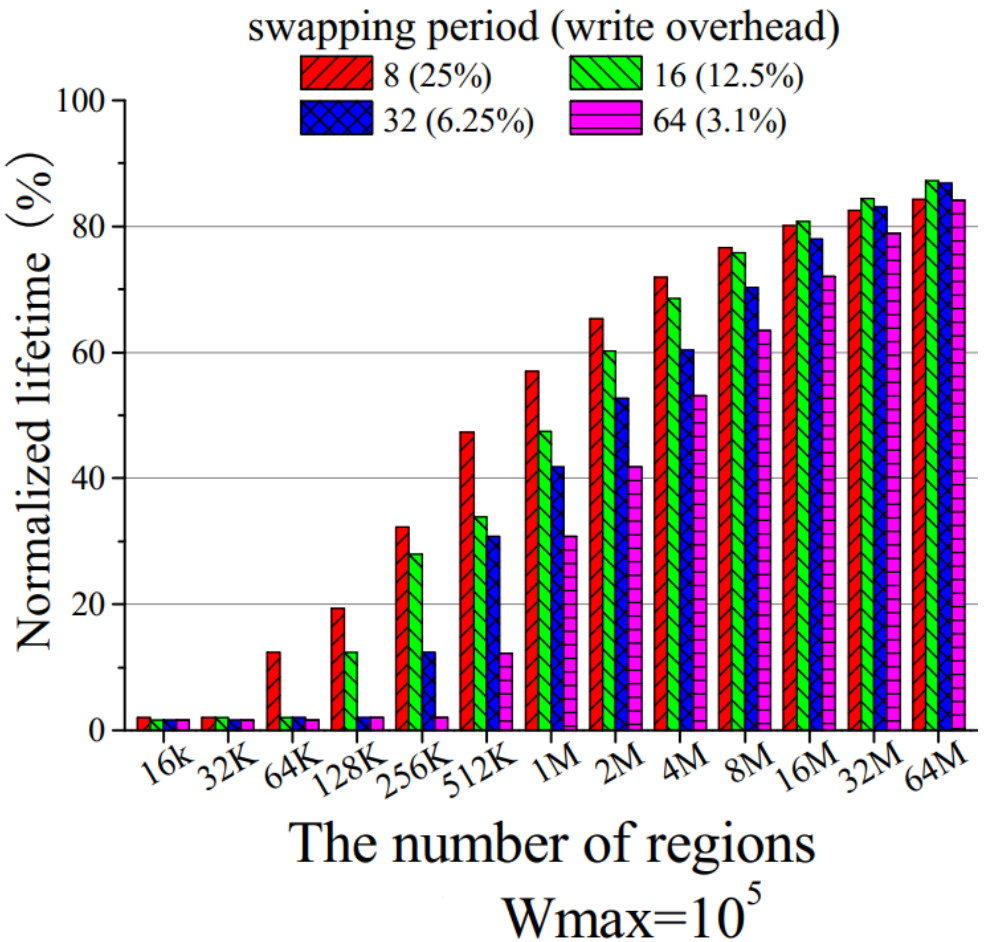




BPA Attack for HWL

➢ The lifetime of HWL under BPA attack





- Smaller wear-leveling granularity increases the NVM lifetime

Problems of Existing Wear-leveling Schemes

- **TBWL and AWL fail to defend against attacks**
  - RAA attack leads to low lifetime in TBWL
  - BPA attack leads to low lifetime in AWL

Problems of Existing Wear-leveling Schemes

- **TBWL and AWL fail to defend against attacks**

- **HWL obtains high lifetime with small granularity**
  - The large granularity leads to low lifetime
  - The small granularity leads to high lifetime

Problems of Existing Wear-leveling Schemes

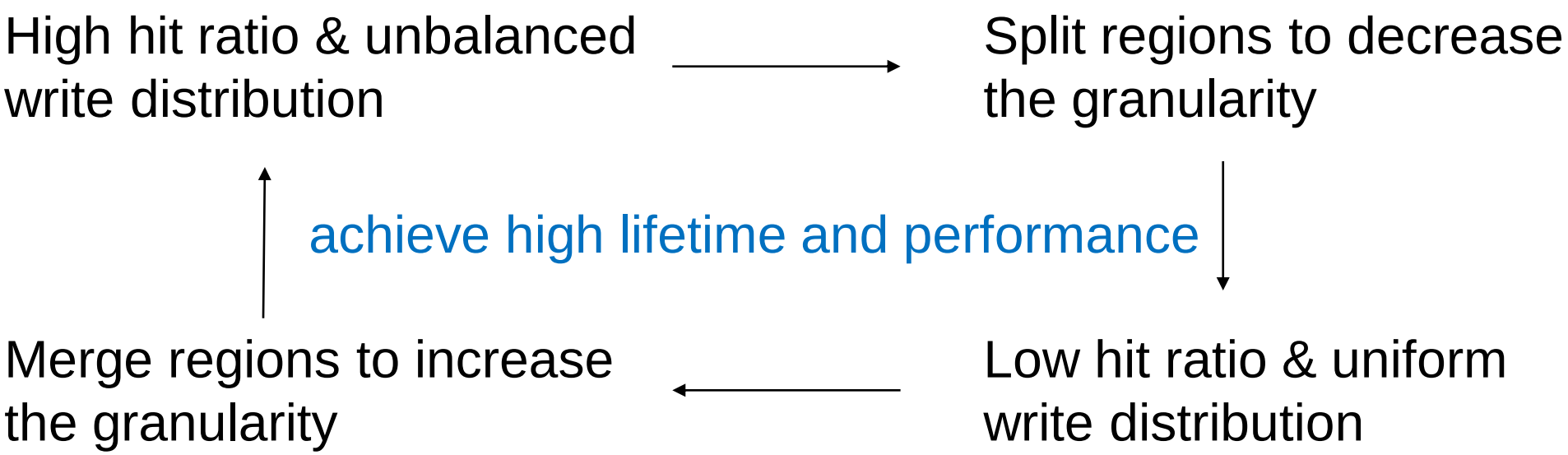
- **TBWL and AWL fail to defend against attacks**
- **HWL obtains high lifetime with small granularity**
- **The cache hit ratio of HWL is affected by the granularity**
  - The wear-leveling entries stored on cache are limited
  - Entries with large granularity cover large NVM ⟶ high cache hit ratio
  - Entries with small granularity cover small NVM ⟶ low cache hit ratio

Problems of Existing Wear-leveling Schemes

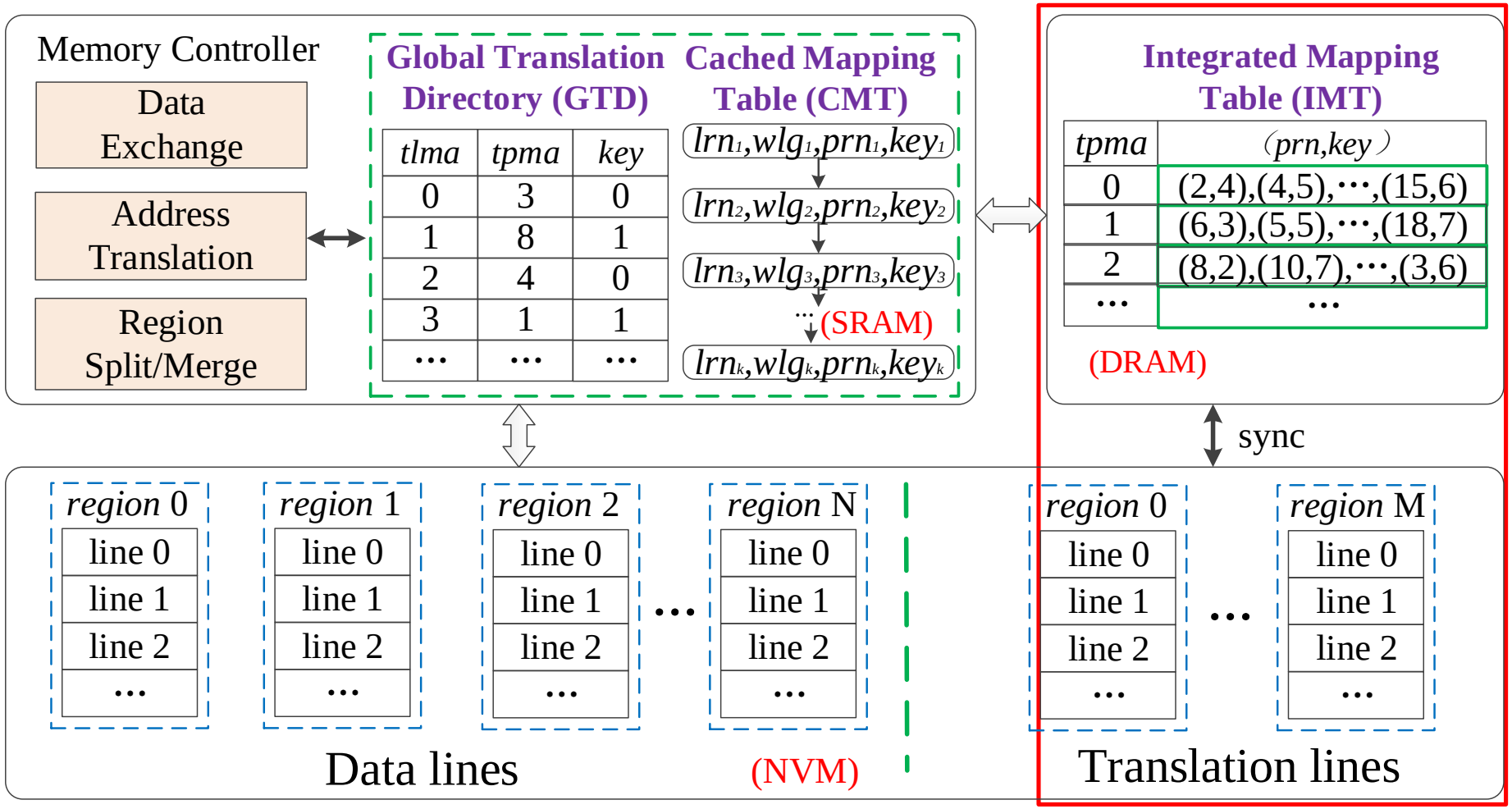
- **TBWL and AWL fail to defend against attacks**
- **HWL obtains high lifetime with small granularity**
- **The cache hit ratio of HWL is affected by the granularity**
- **High performance and lifetime are in conflict**

How to address the conflict between the lifetime and performance is important

SAWL: self-adaptive wear-leveling scheme



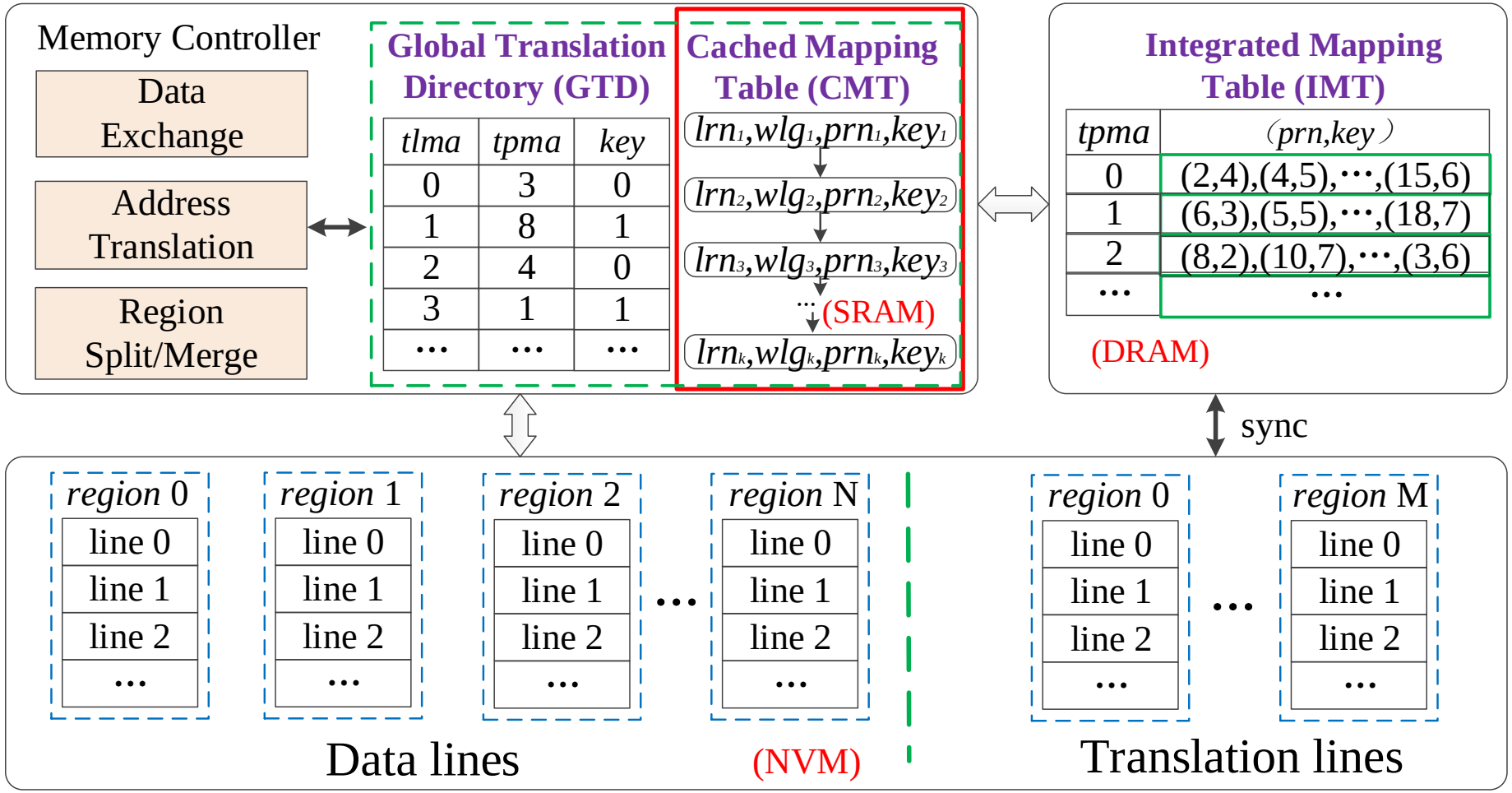
Architecture of SAWL



- IMT (translation lines): record the locations in which the user data are actually stored

  ⟶ wear-leveling the data lines

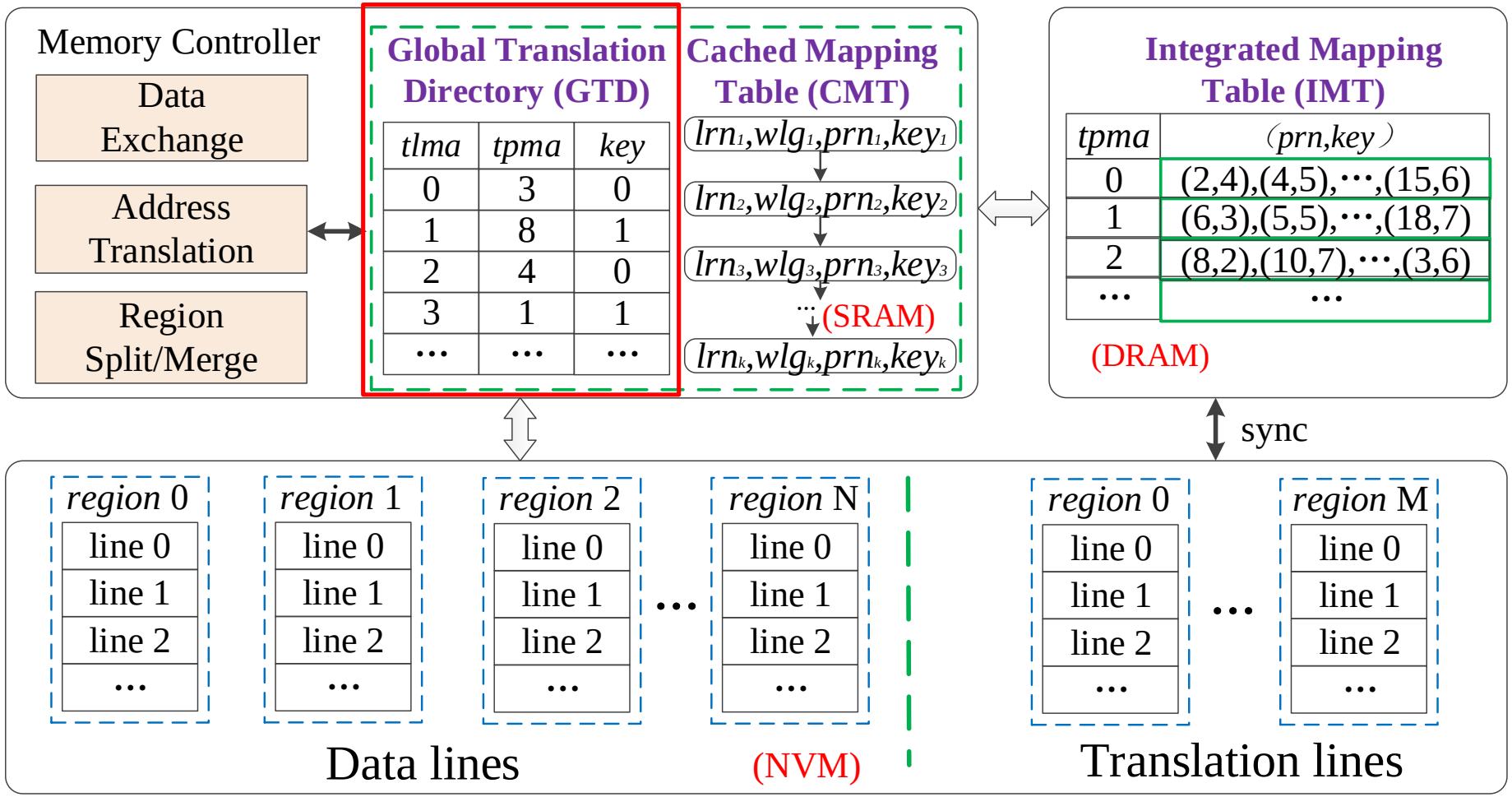
Architecture of SAWL



- CMT: buffer the recently used IMT entries

⟶ reduce translation latency

Architecture of SAWL



- GTD: record the relationships between logical/physical addresses of translation lines
  ⟶ wear-leveling the translation lines

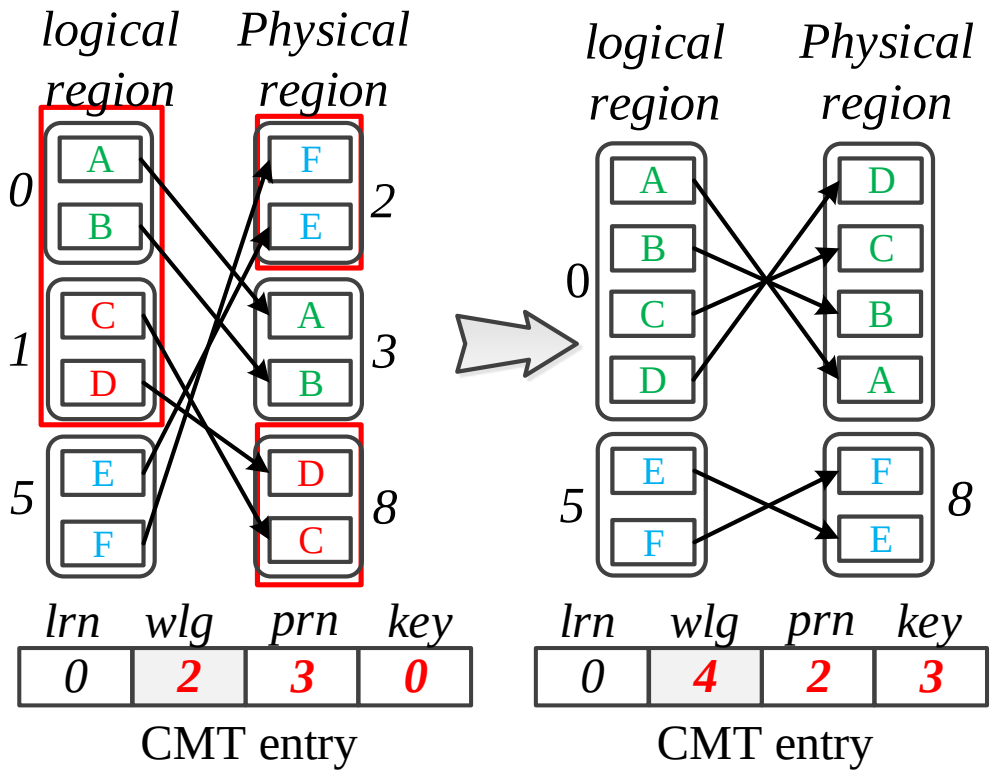
Operations in SAWL

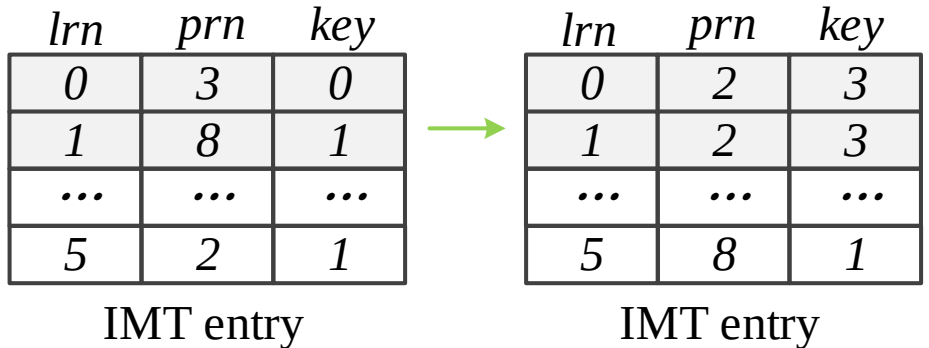
➢ Merge the region

1. Choose two unmerged neighborhood logical regions.

2. Physically exchange the data to satisfy the algebraic mapping between the logical and physical regions.

3. Update the relevant CMT entries on the SRAM and the IMT table on the NVM.



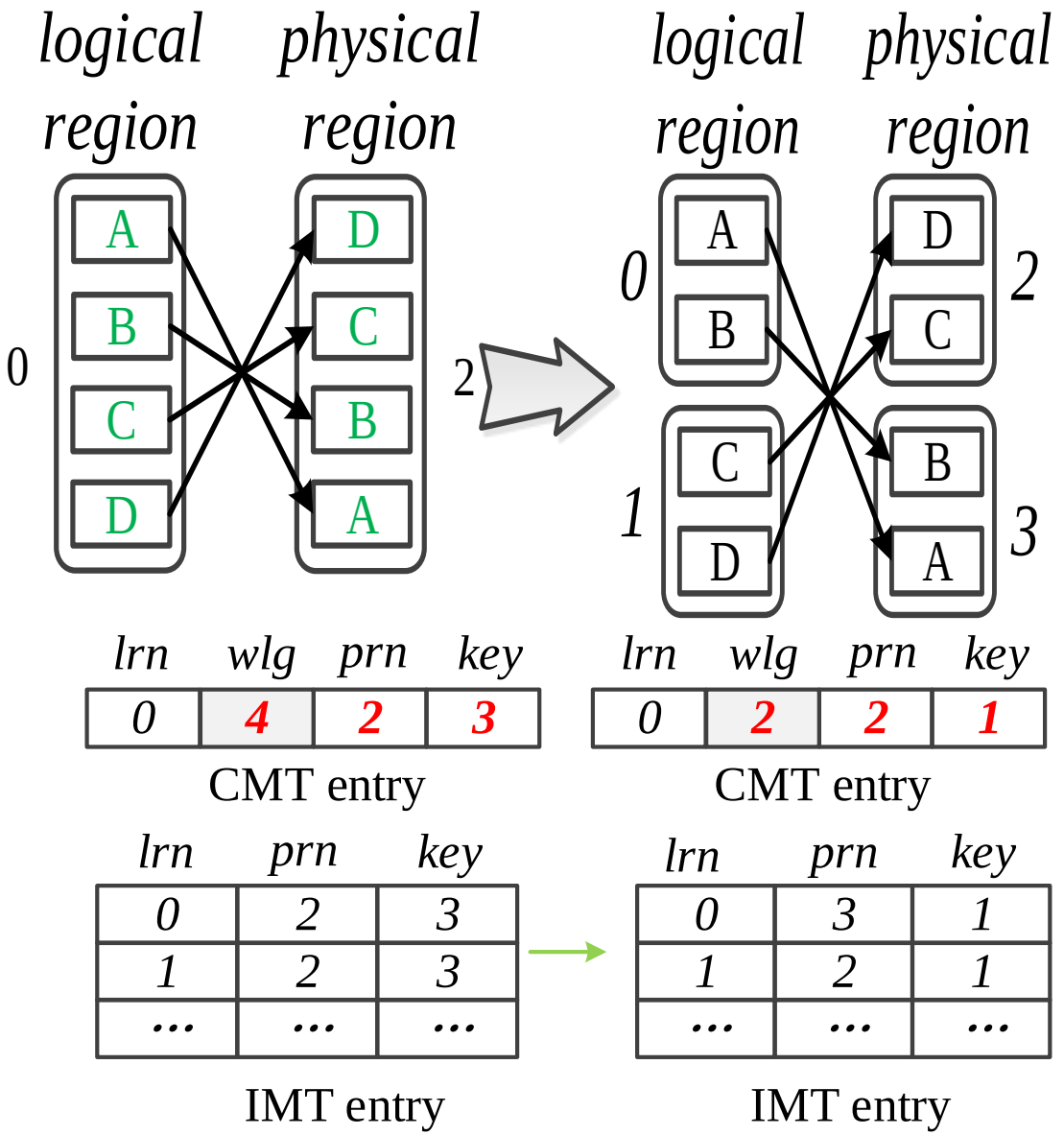


Operations in SAWL

- **Merge the region**
- **Split the region**

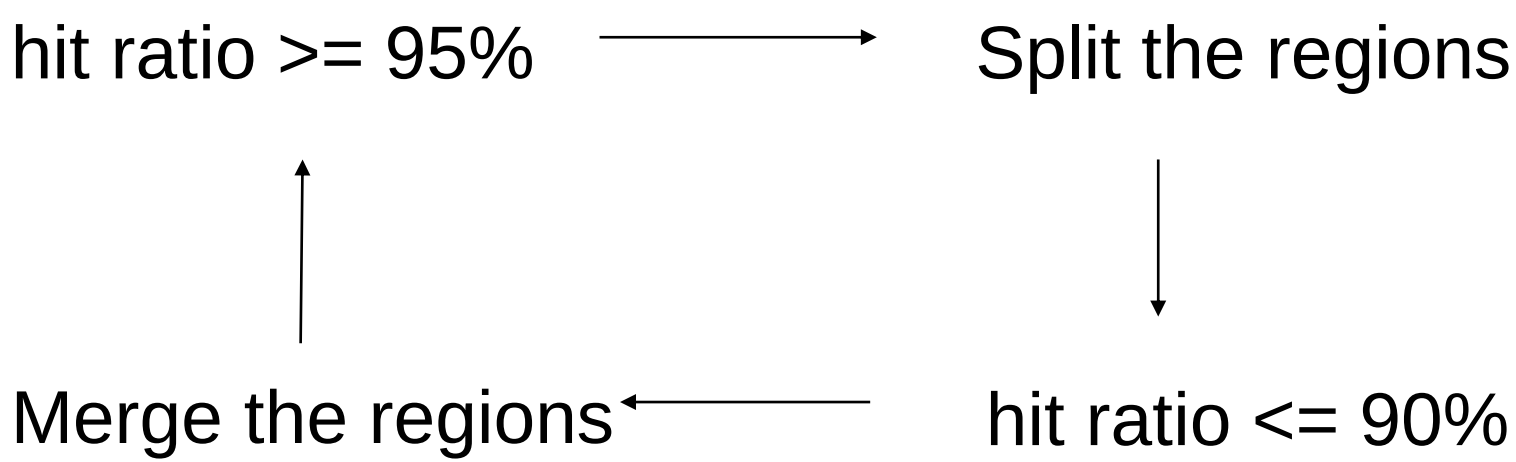
1. Logically split the region without data move. The data have already satisfied the algebraic mapping.

2. Update the CMT entries and IMT table.



When to Adjust the Region

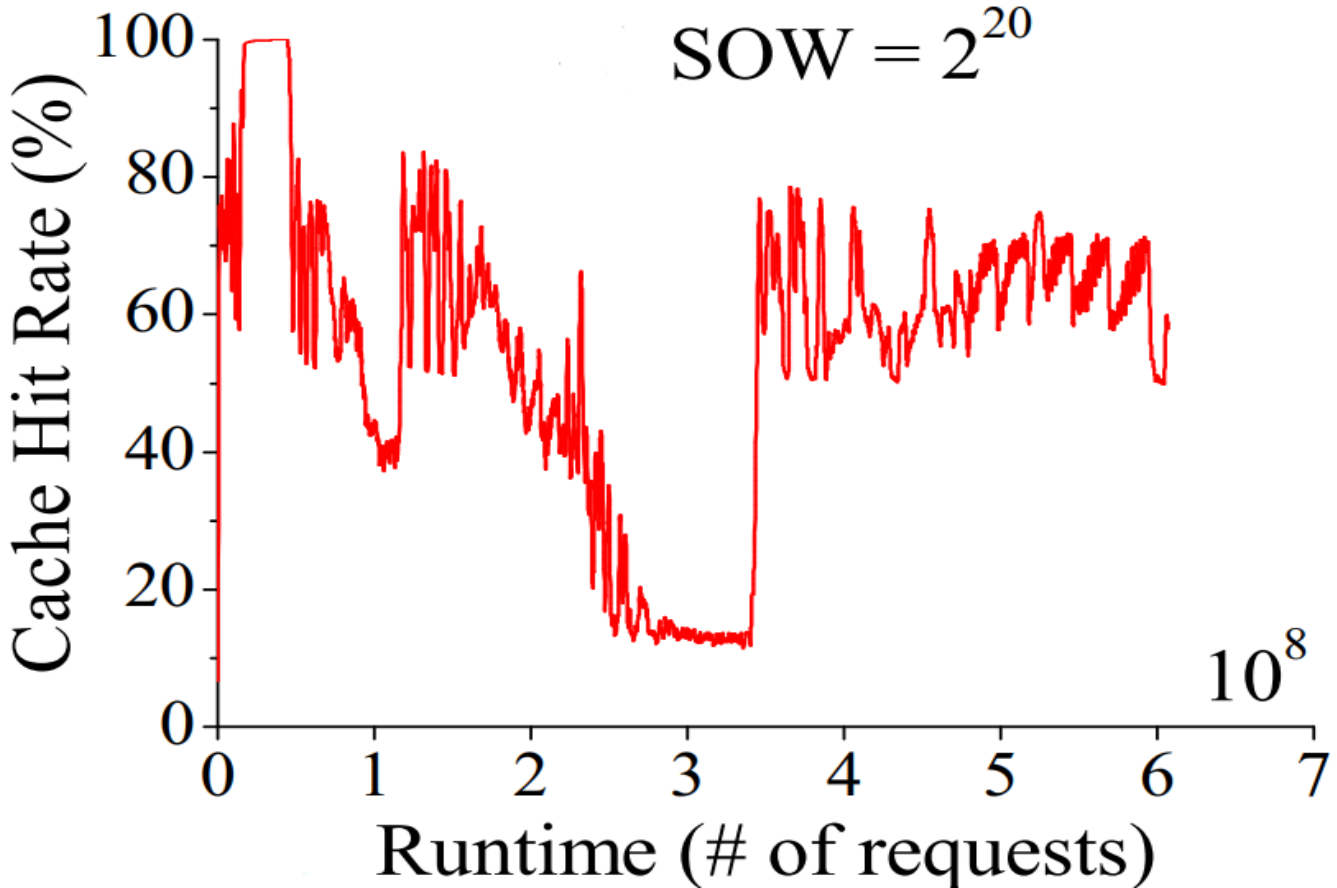
- **We use the hit ratio as the trigger to merge/split region**
  - Hit ratio below 90% significantly decreases the performance
  - Hit ratio above 95% slightly impacts on the performance



Parameter in SAWL

➢SOW: size of the observation window

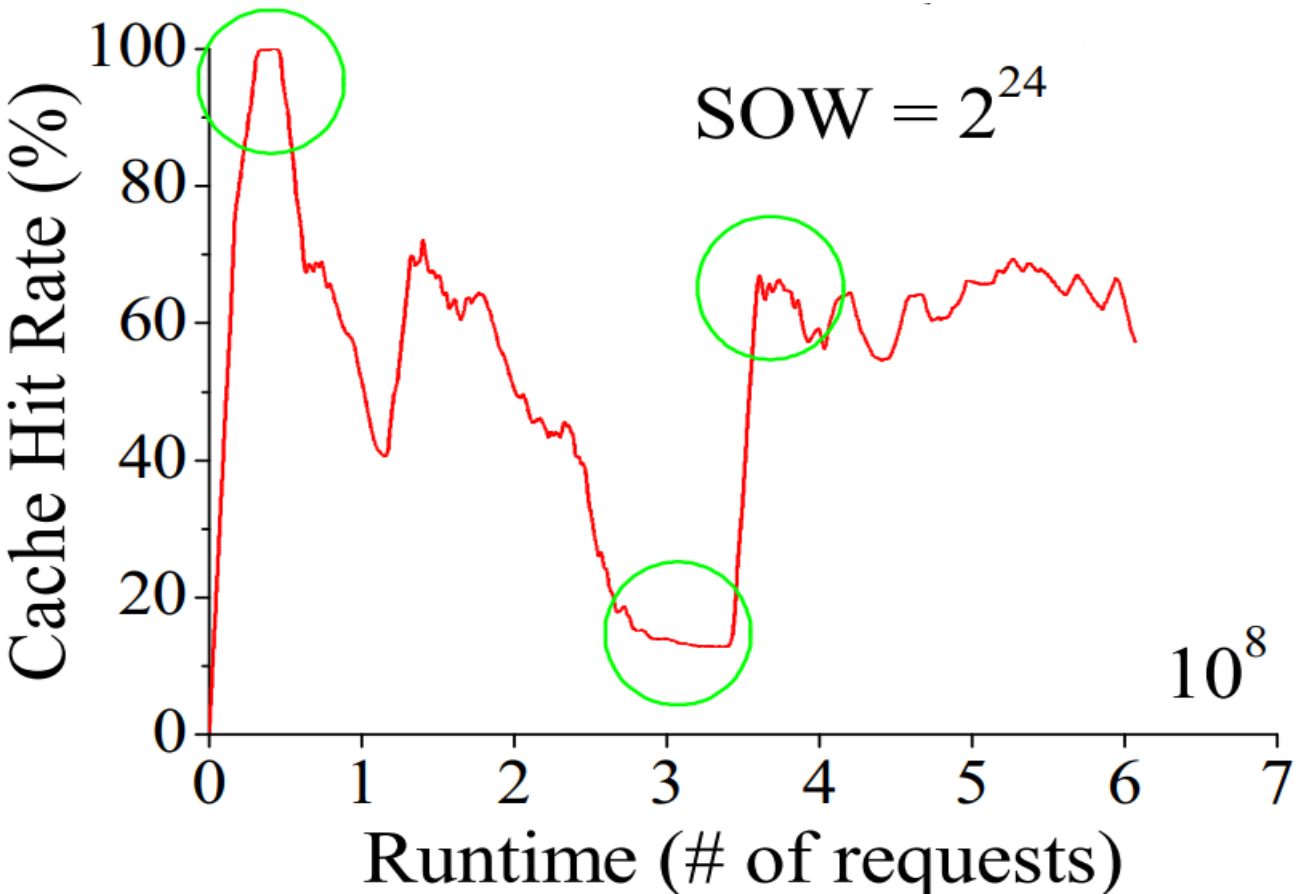
- Small SOW -> cache hit rate frequently fluctuates



Parameter in SAWL

➢SOW: size of the observation window

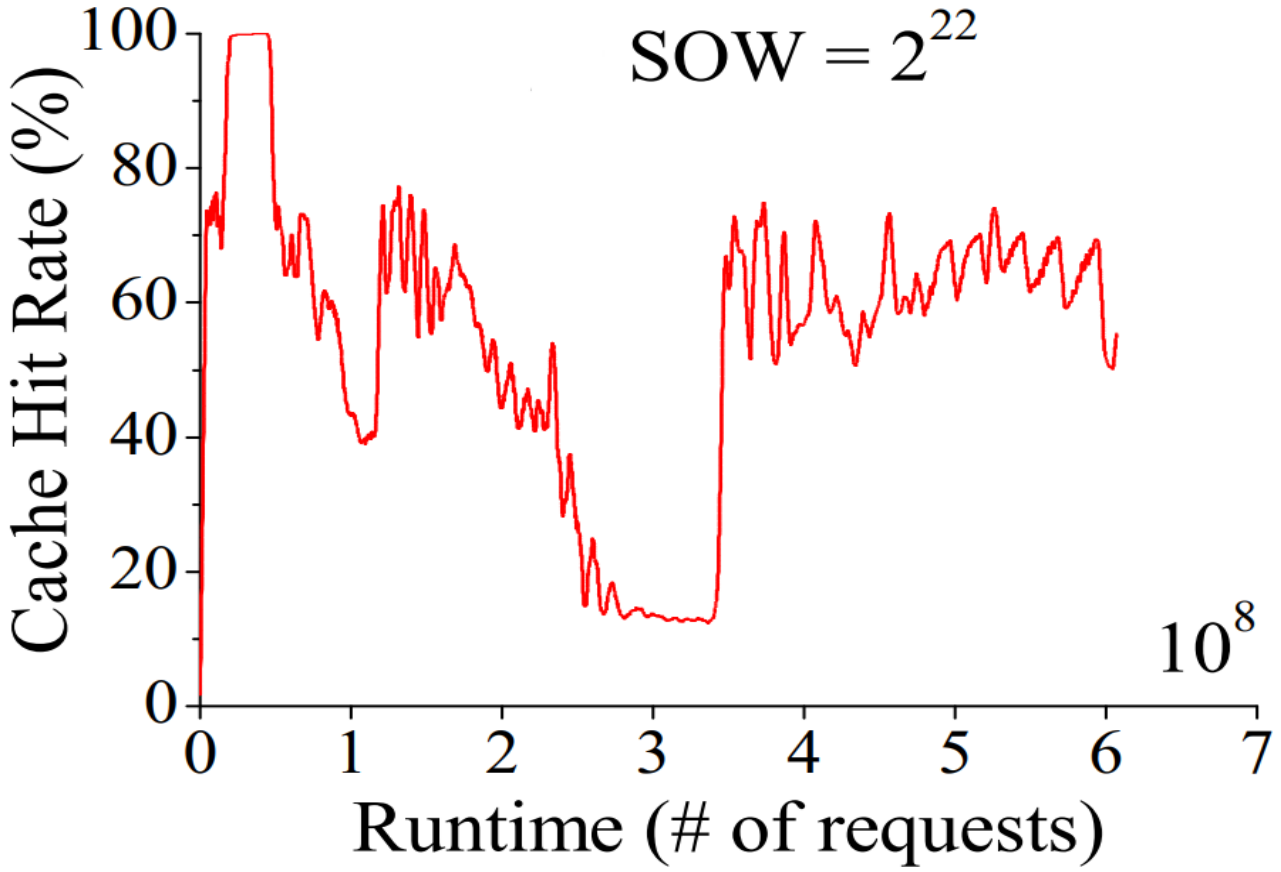
- Small SOW -> cache hit rate frequently fluctuates
- Large SOW -> systems miss important trigger point



Parameter in SAWL

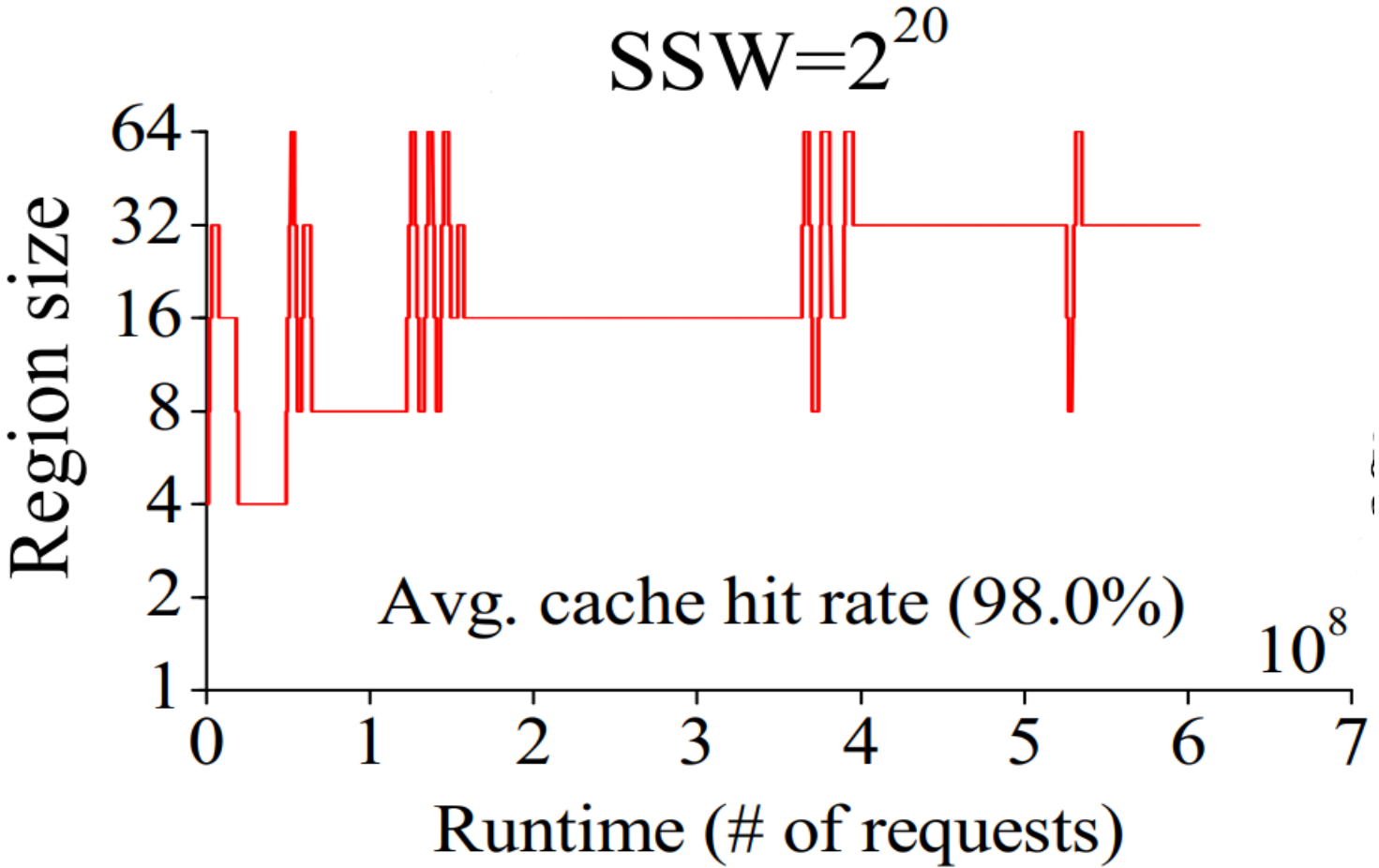
➢SOW: size of the observation window

- Small SOW -> cache hit rate frequently fluctuates
- Large SOW -> systems miss important trigger point
- We use 2^22 as the SOW value



Parameter in SAWL

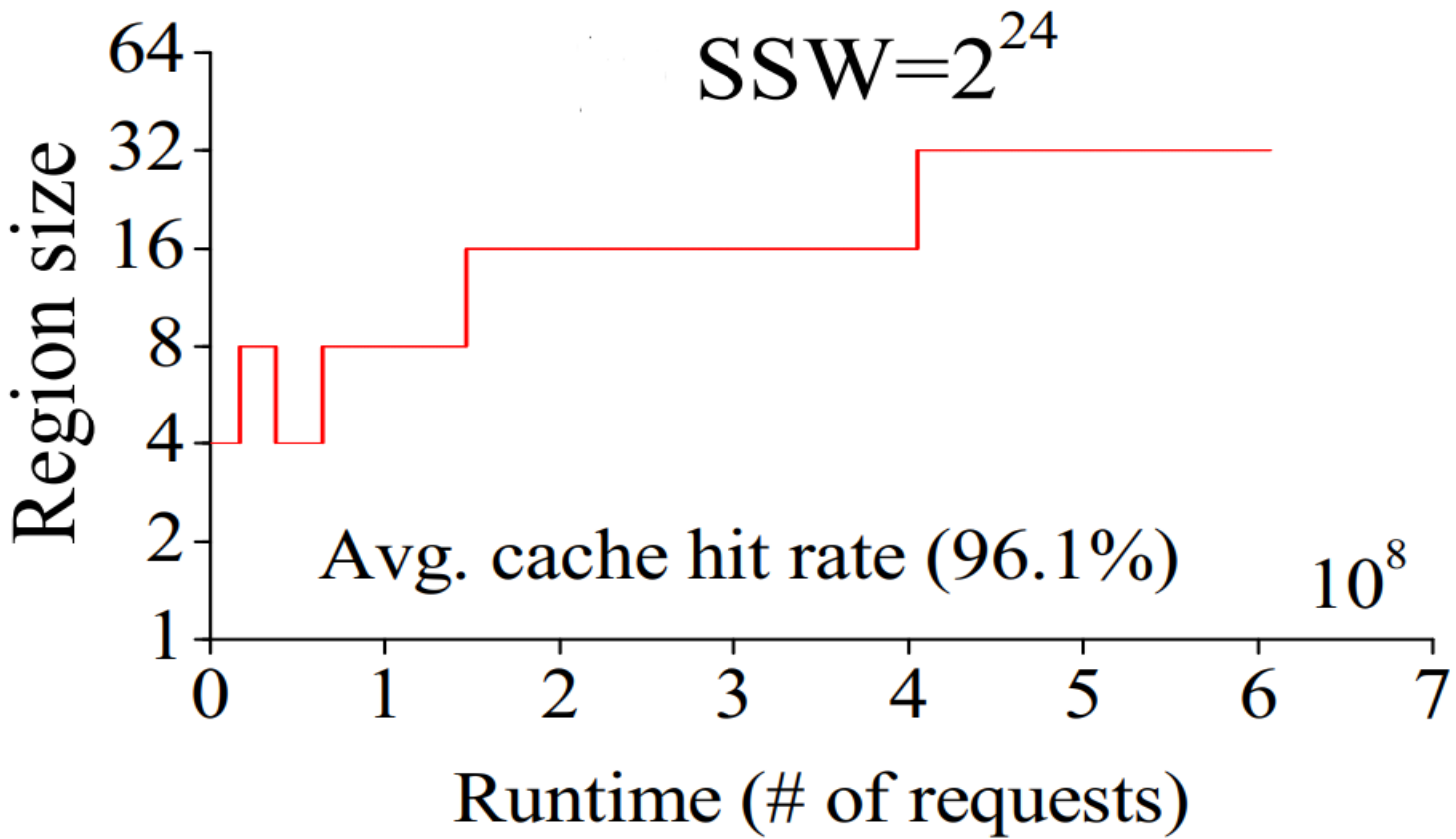
- **SSW: size of the settling window**
  - Small SSW -> frequently adjust region size



Parameter in SAWL

➢SSW: size of the settling window

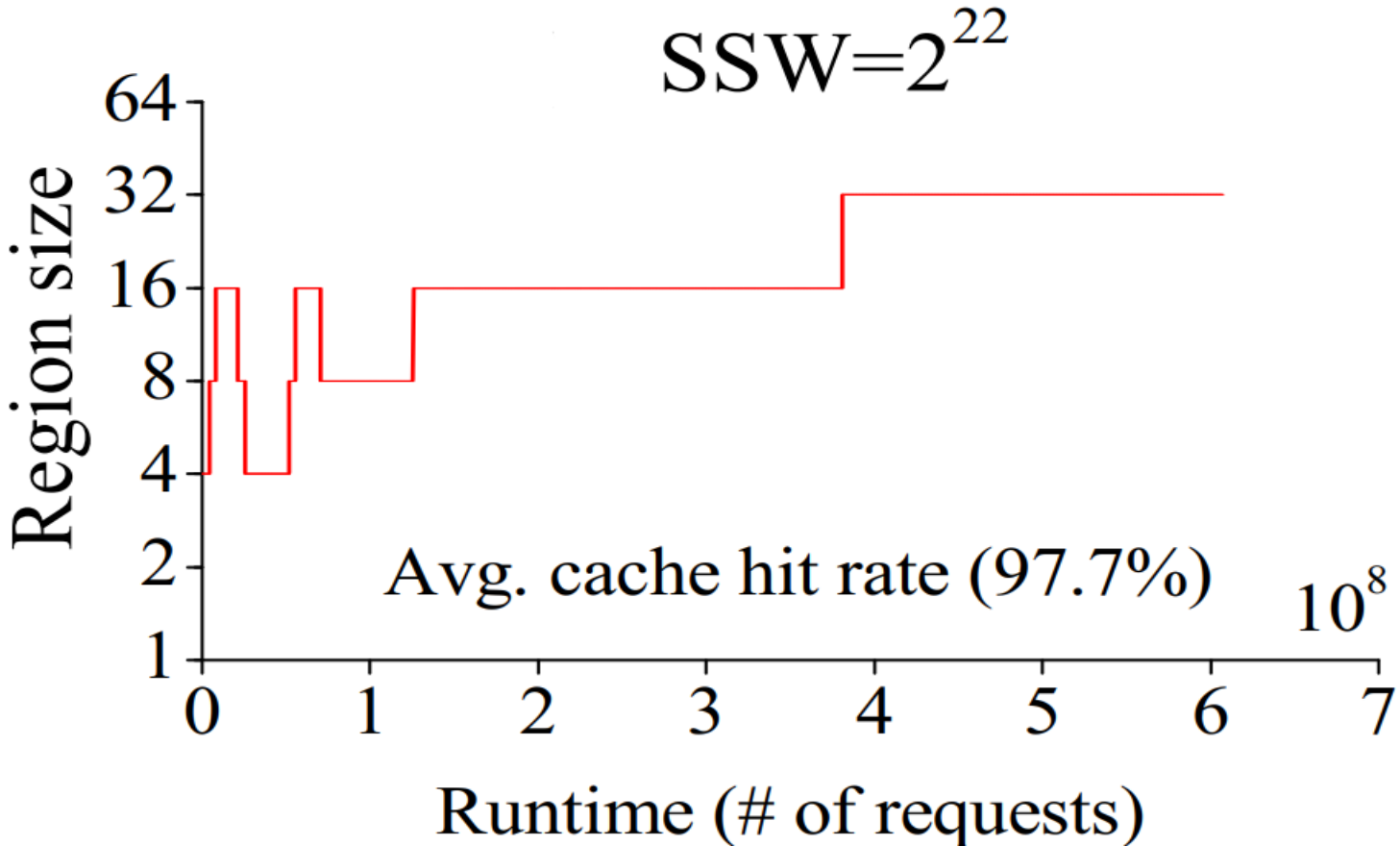
- Small SSW -> frequently adjust region size
- Large SSW -> fail to sufficiently adjust the region size



Parameter in SAWL

➢SSW: size of the settling window

- Small SSW -> frequently adjust region size
- Large SSW -> fail to sufficiently adjust the region size
- We use 2^22 as the SSW value



Experimental Setup

➢Configuration of simulated system via Gem5

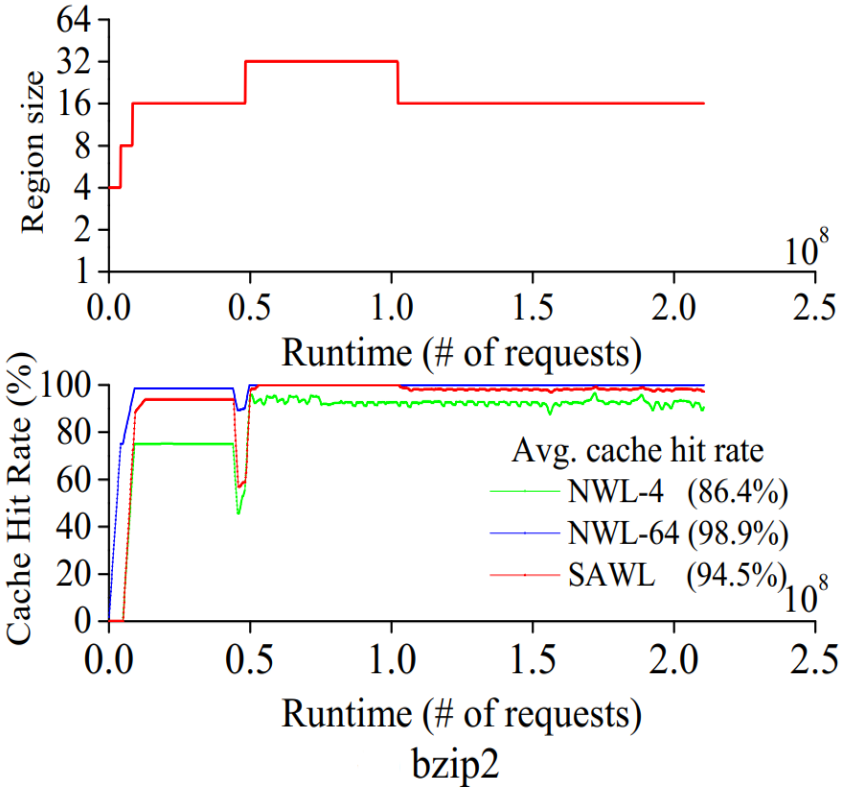
CPU	8 cores, X86-64 processor, 3.2 GHz
Private L1 cache	64KB
Shared L2 cache	512KB
CMT cache	256KB
DRAM/PCM Capacity	128MB/8GB
Read/Write latency model	DRAM 50/50ns, PCM 50/350ns
Address translation latency	Cache hit 5ns, Cache miss 55ns

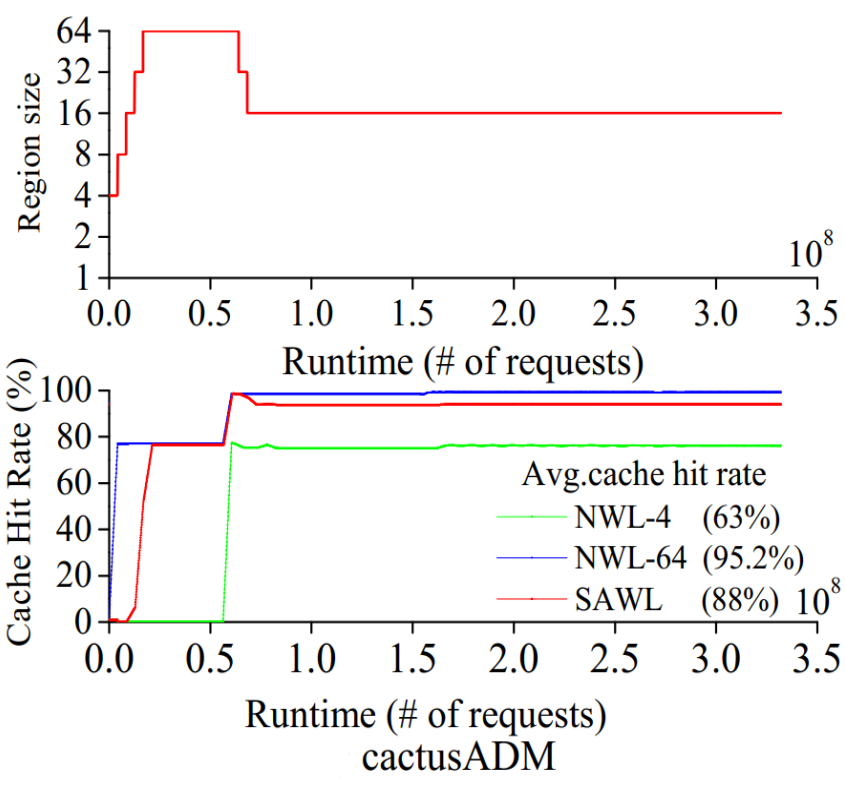
➢Comparisons

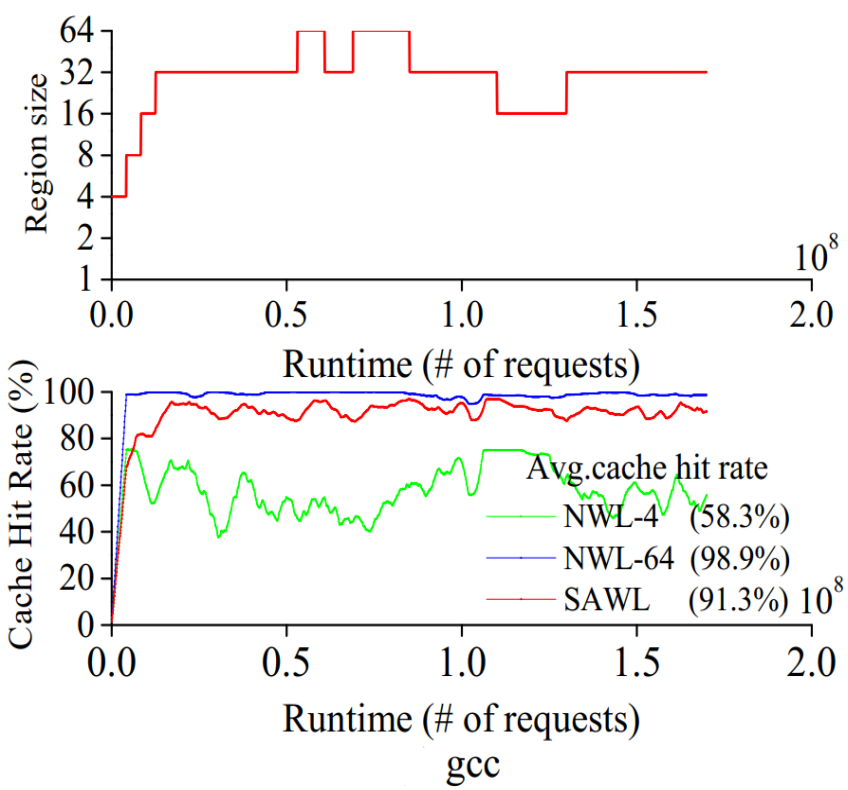
- Baseline: an NVM system without wear-leveling scheme.
- NWL-4: naive wear-leveling scheme with a region consisting of 4 memory lines.
- NWL-64: naive wear-leveling scheme with a region consisting of 64 memory lines.
- AWL schemes: RBSG and TLSR.
- HWL schemes: PCM-S and MWSR.

➢Benchmark: SPEC2006

Cache Hit Rate

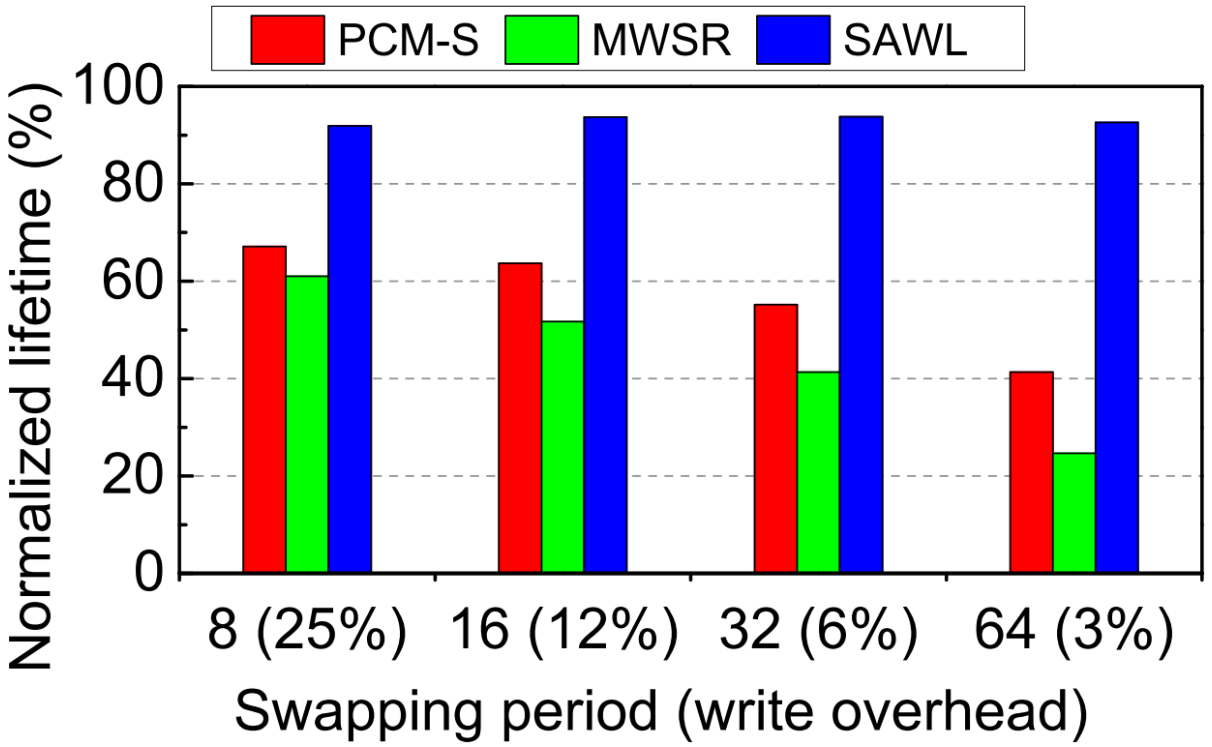




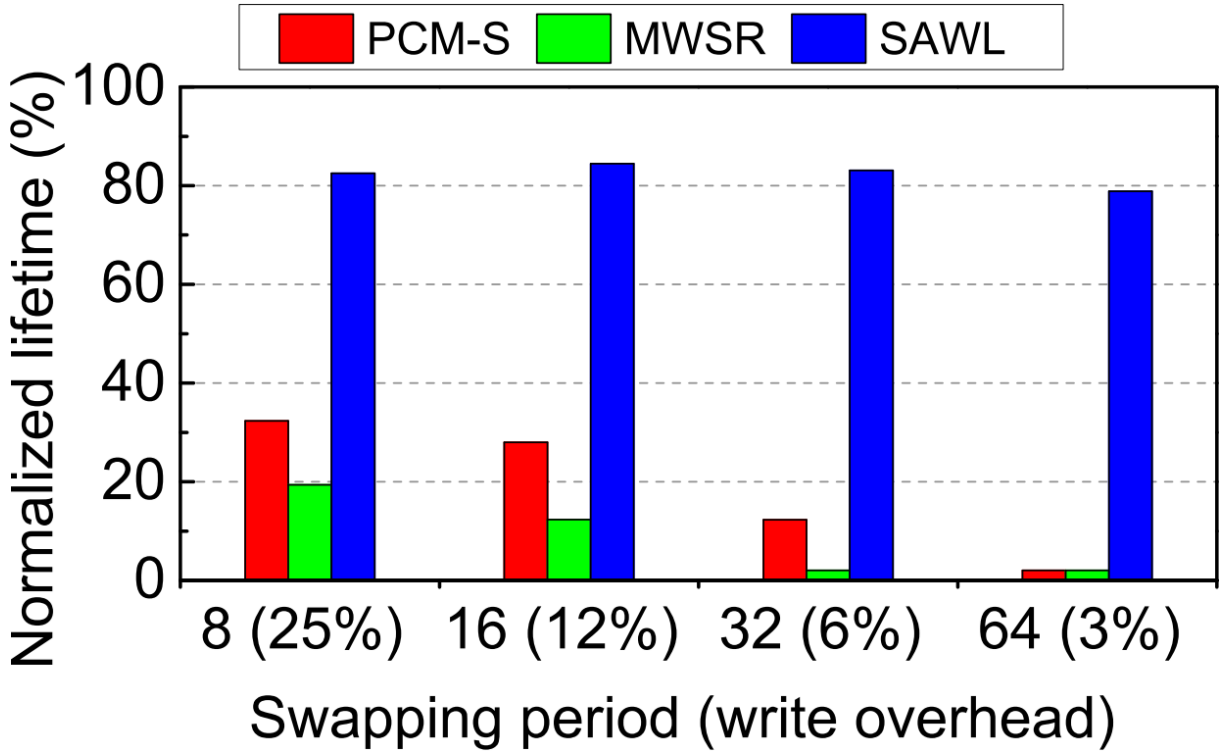


- SAWL has high cache hit rate, and adjusts the region size according to the hit rate.

Lifetime under BPA Program



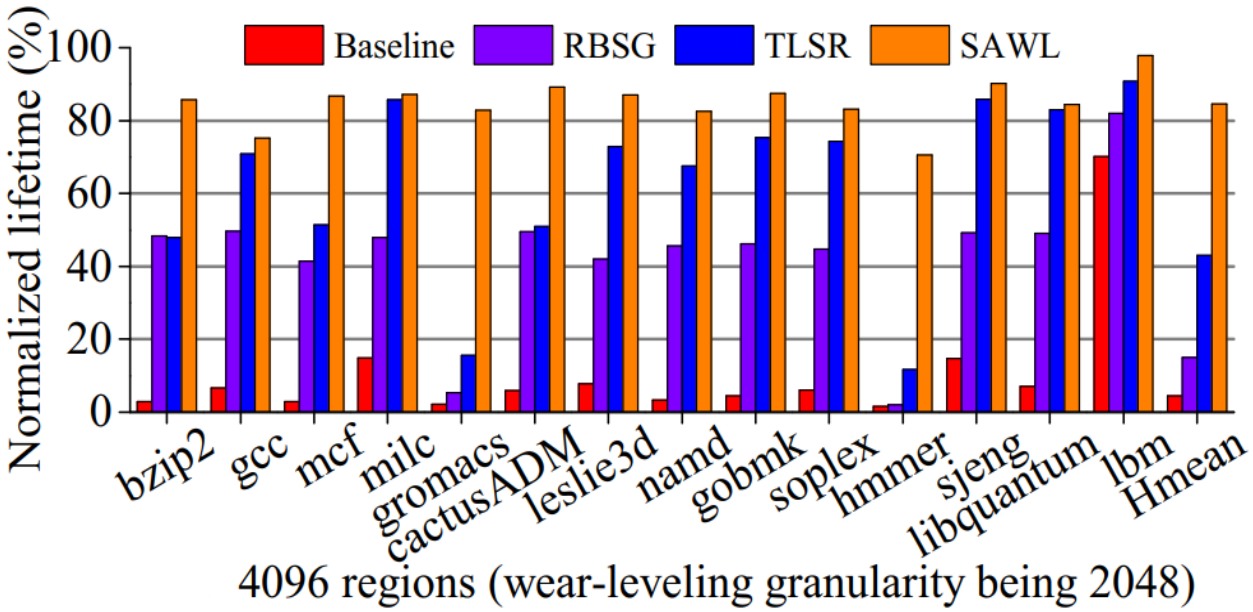
10^6 endurance



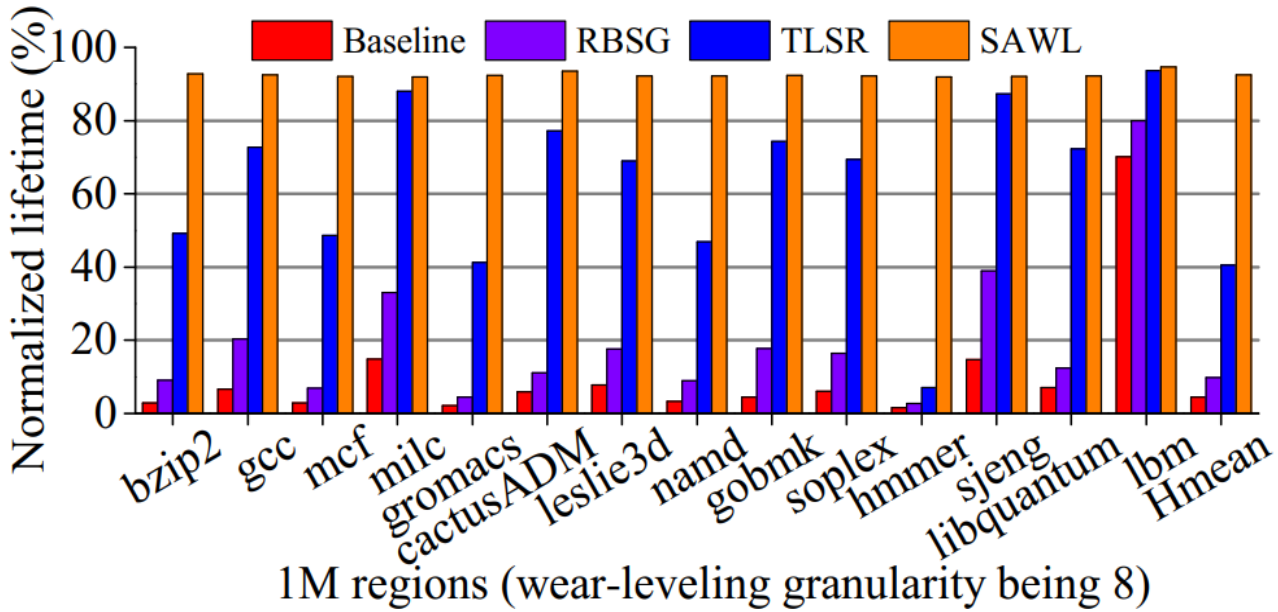
10^5 endurance

- Smaller swapping period increases the NVM lifetime at the cost of high write overhead.
- SAWL efficiently defends against the BPA attack.

Lifetime under Applications



4096 regions (wear-leveling granularity being 2048)



1M regions (wear-leveling granularity being 8)

- SAWL achieves high lifetime in all applications.

Conclusion

- Existing wear-leveling schemes fail to efficiently work on MLC NVM
- SAWL dynamically tunes the wear-leveling granularity
  - Low cache hit ratio with uniform write distribution leads to merging regions
  - High cache hit ratio with unbalanced write distribution leads to splitting regions
- SAWL achieves high lifetime under attacks and general applications with low performance overhead

Thanks! Q&A

Email: jmhuang@hust.edu.cn