UKSM: Swift Memory Deduplication via Hierarchical and Adaptive Memory Region Distilling

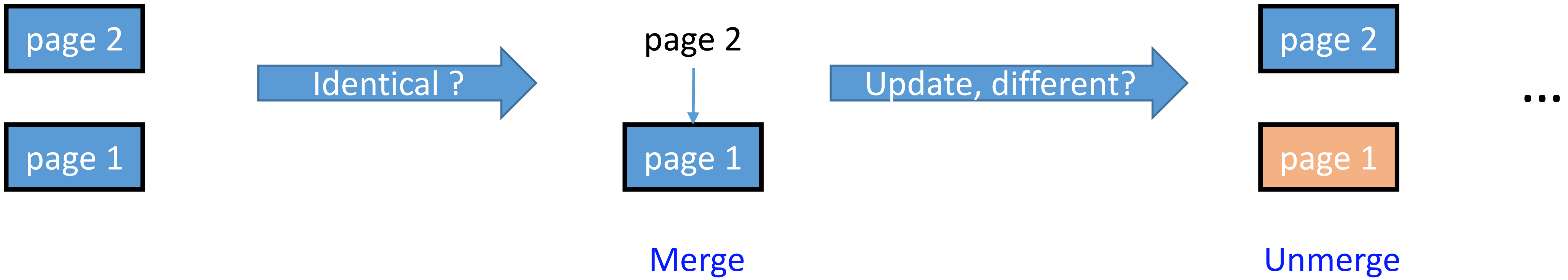
Nai Xia*   Chen Tian*   Yan Luo+   Hang Liu+   Xiaoliang Wang*

*: Nanjing University   +: University of Massachusetts Lowell

Feb/15/2018

Background

- What is **K**ernel **S**amepage **M**erging (KSM)?



- Goal: Reduce memory consumption when duplication exists.
- Effectiveness: There exist tremendous (~86%) memory duplications in real-world applications, Change *et al.* [ISPA 2011].

Unique Challenges

- Storage deduplication deals with relatively **static** content, only concerns about **duplication ratio**.
  - Sparse Indexing [FAST 2009] , CAFTL [FAST 2011], El-Shimi *et al.* [ATC 2012], Cao *et al.* [Just now]

- Responsiveness:
  - Remove duplications before they exhaust the memory.

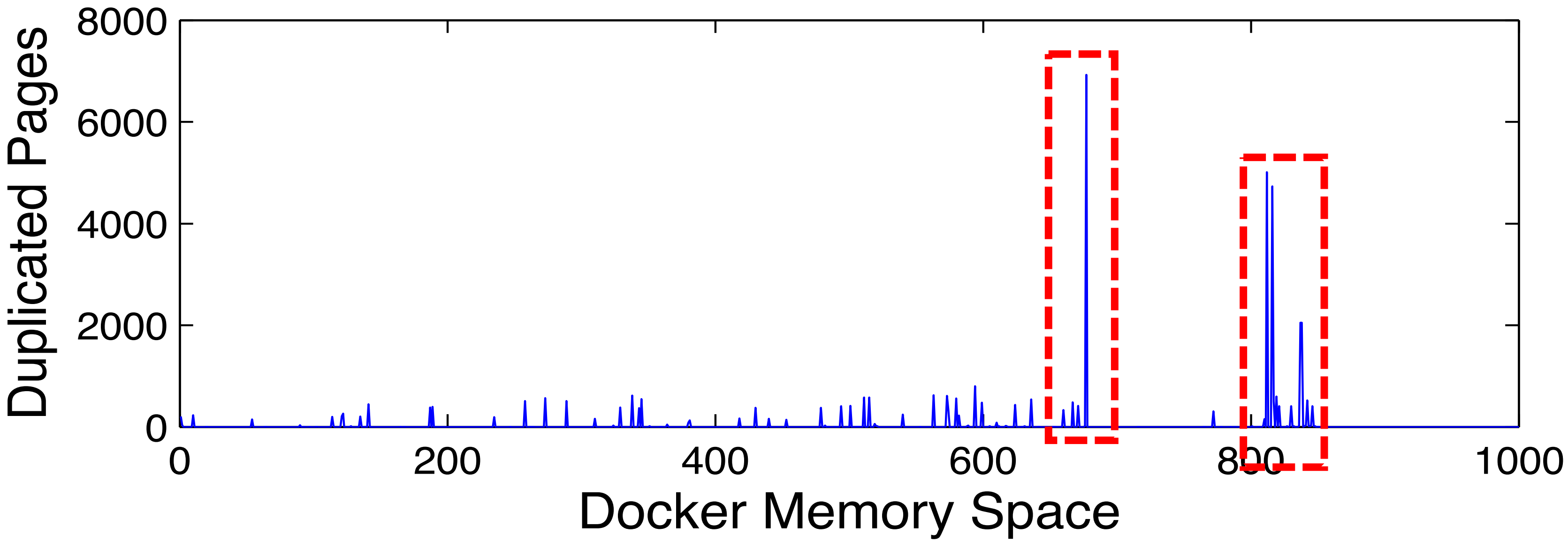
- Dynamic nature:
  - Duplication status may change over time.

Accelerate the deduplication of memory which is dynamic in nature!

Outline

- Observation (Opportunity)
- Overview
- Hierarchical Region Distilling
- Adaptive Partial Hashing
- Evaluation
- Conclusion

Observation I: Pages within the Same Region Present Similar Patterns



*Please refer to our paper for other pattern analysis

- Test: Apache web server and MySQL database serving wordpress website in Ubuntu 16.04 (kernel version 4.4).

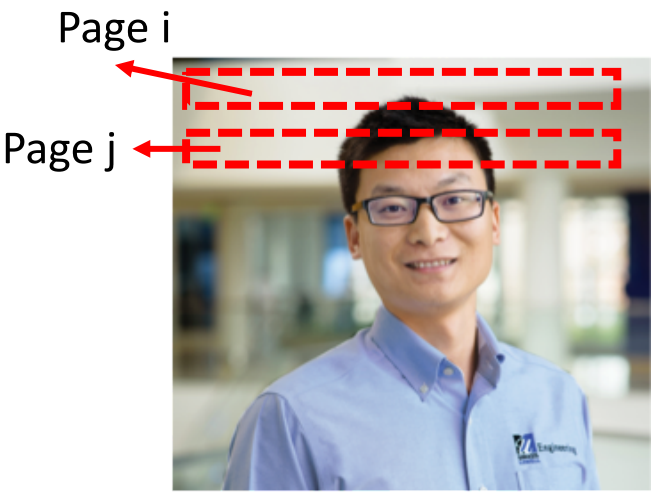
Duplicated pages concentrate by memory region.

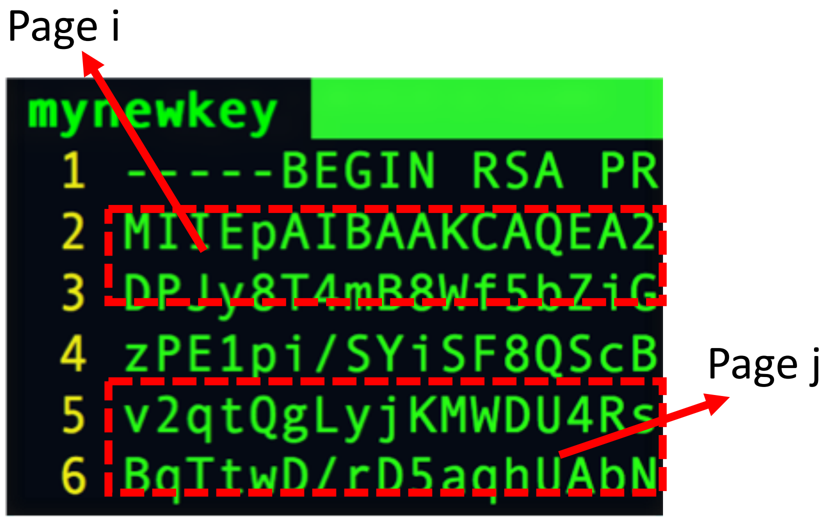
UMASS LOWELL Learning with Purpose

FAST'18

Observation II: Hashing Needs to Be Adaptive

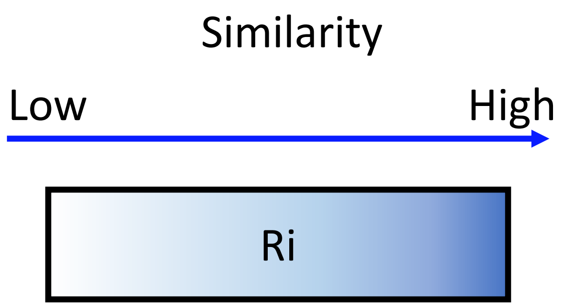
- Various applications need different hashing strengths to differentiate:
  - Image applications contain pages with highly similar contents.
  - Crypto applications contain diverse contents.



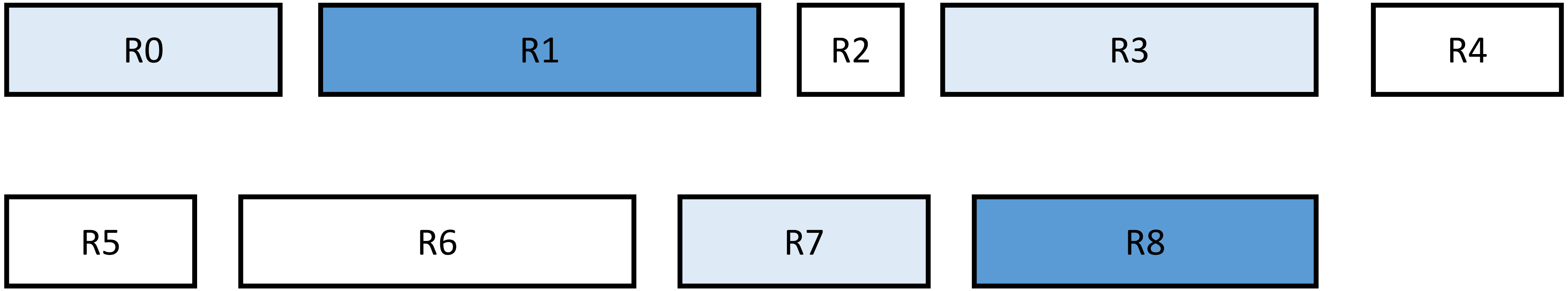


We should adjust hashing strength accordingly.

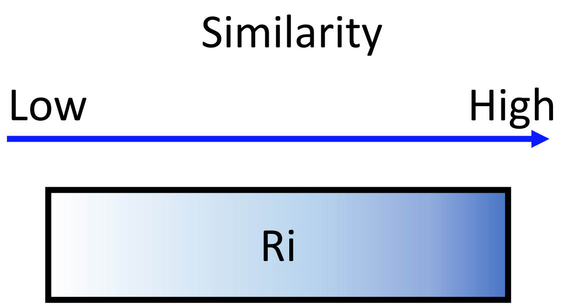
Overview



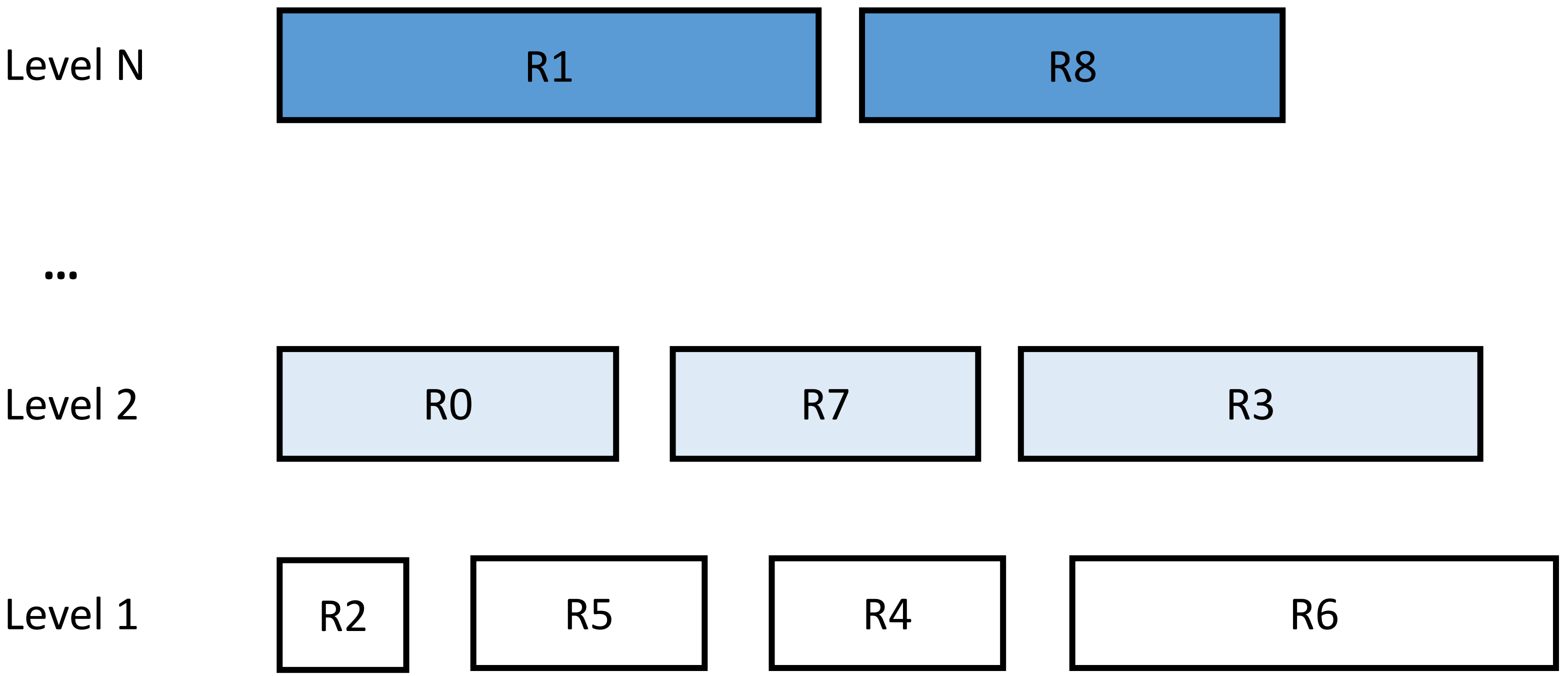
- Assuming we have 9 memory regions, i.e., R0 – R8.



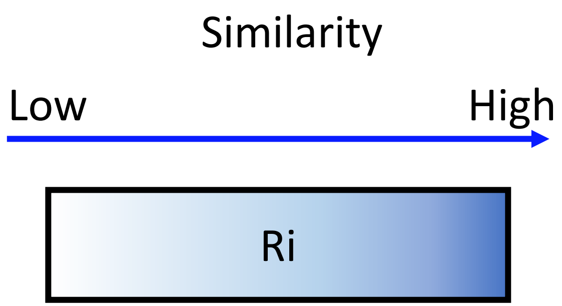
Overview



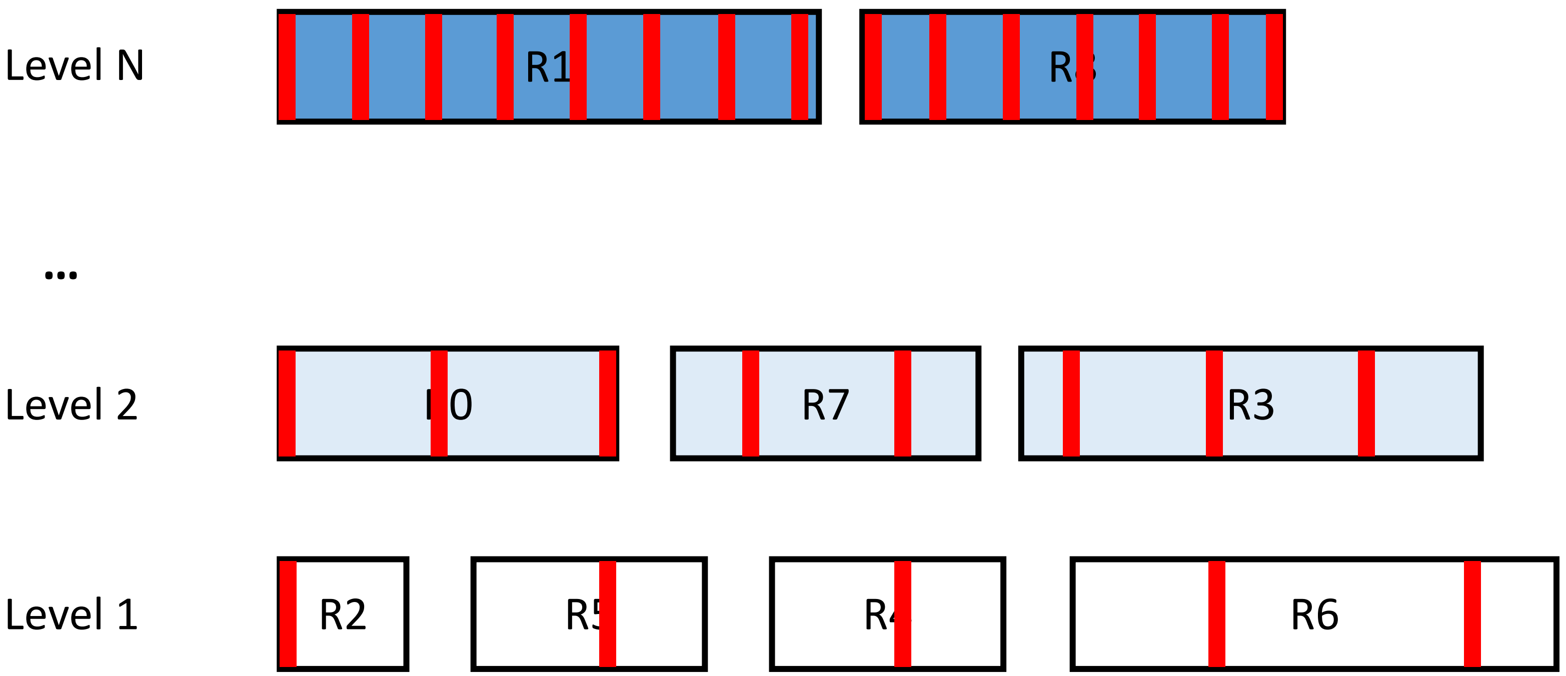
- Hierarchical memory region clustering.



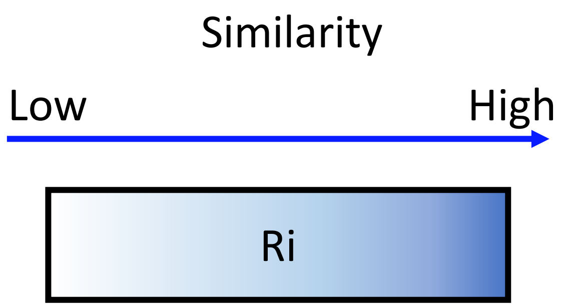
Overview



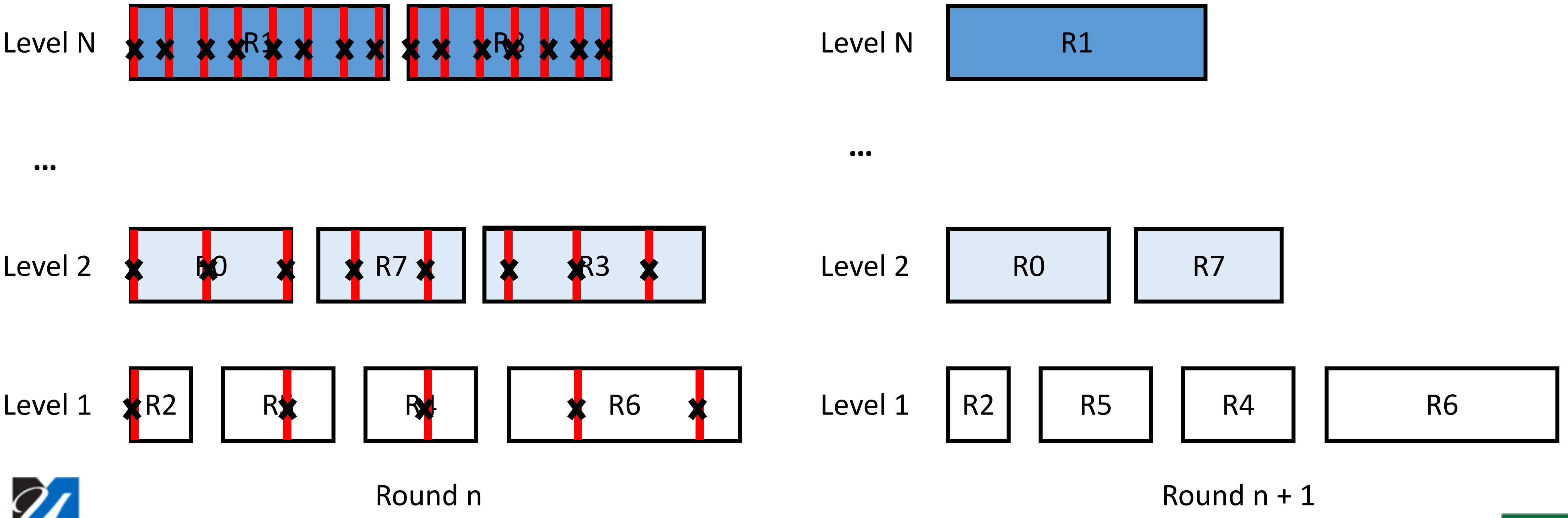
- **Hierarchical region distilling.**



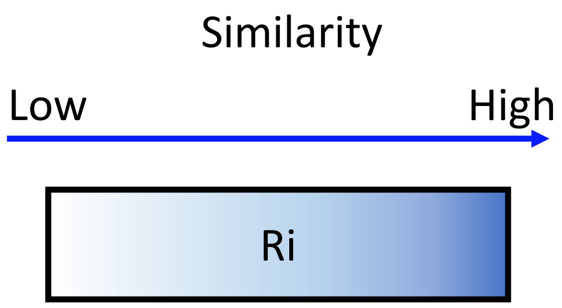
Overview



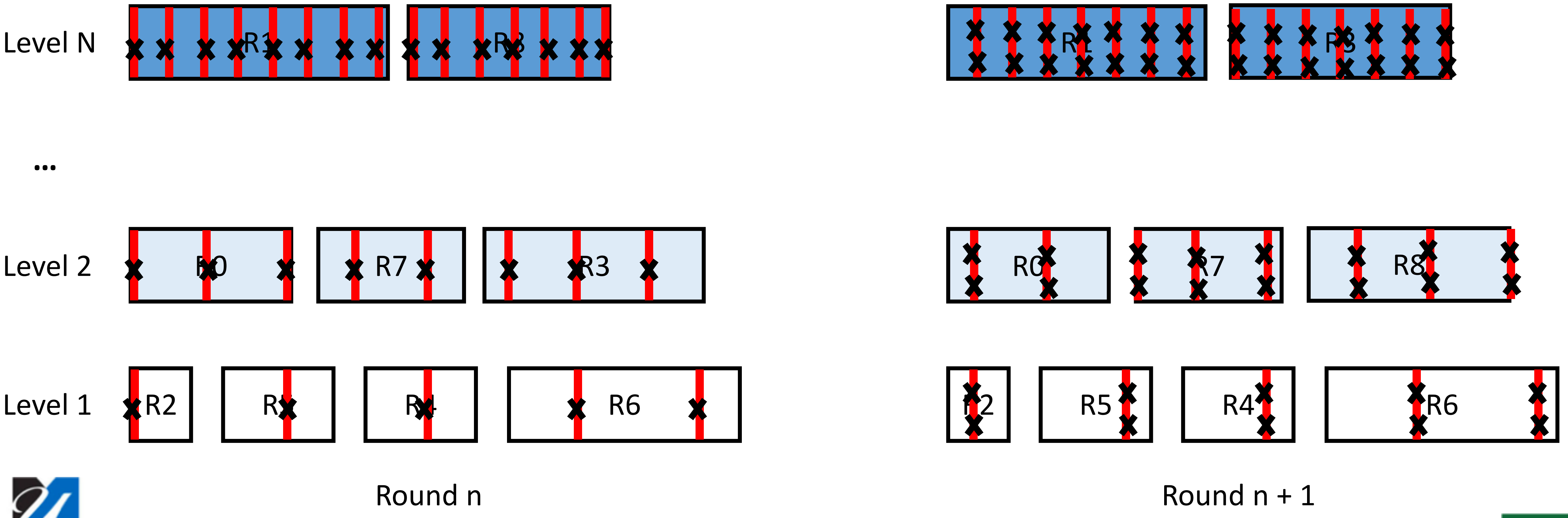
- **Hierarchical region distilling.**



Overview



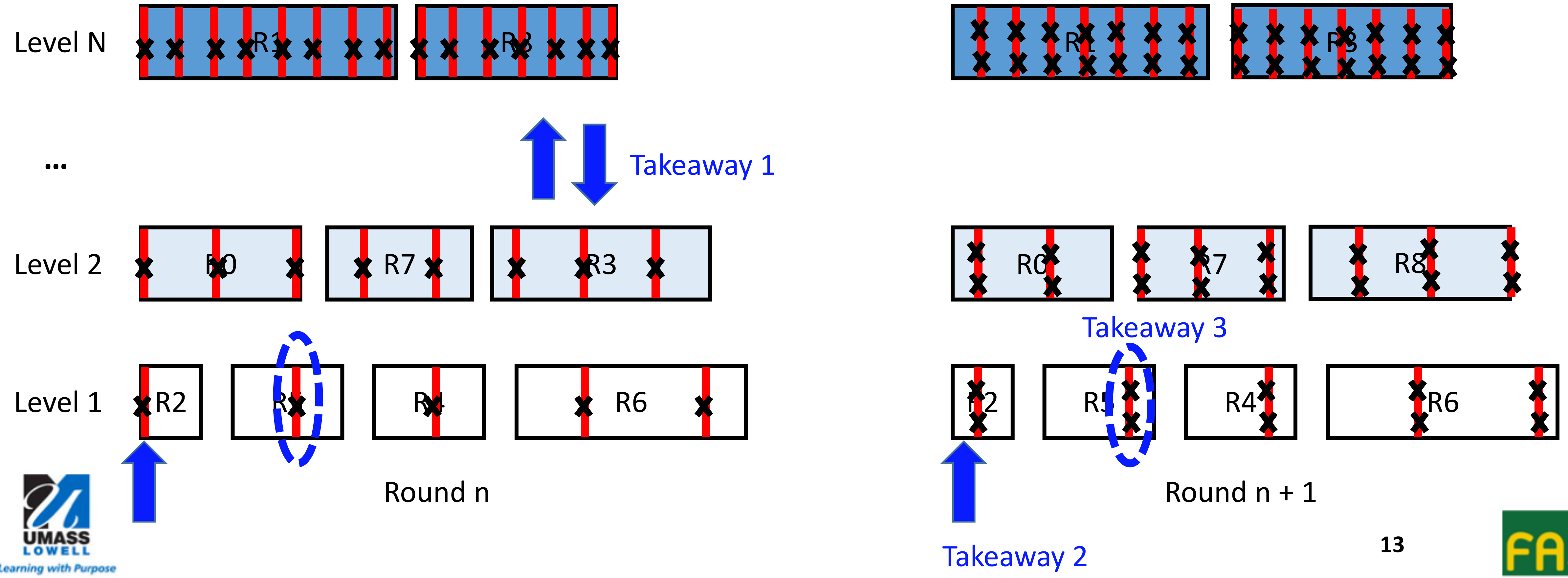
- Hierarchical region distilling + **Adaptive partial hashing.**



Overview

- Takeaway 1: Promote/demote regions.
- Takeaway 2: Sampling offset shift.
- Takeaway 3: Hash strength adjustment.

- Hierarchical region distilling + **Adaptive partial hashing.**



Hierarchical Region Distilling

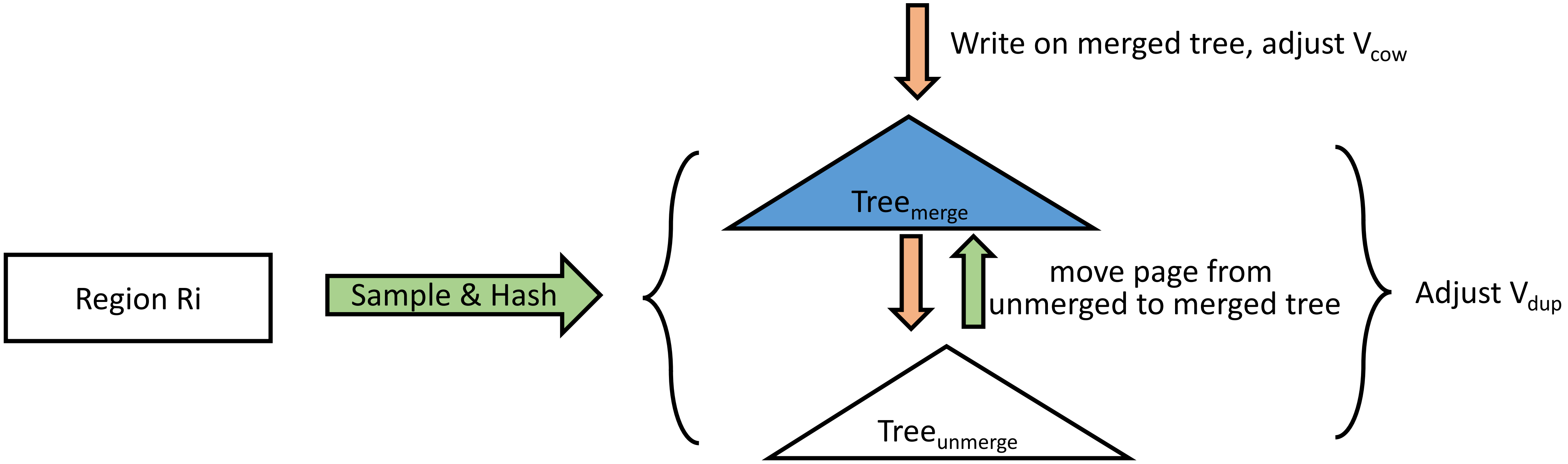
* COW: copy on write

- Memory region characterization – **Signatures**:
  - V_{cow}: promote regions whose COW-broken ratios are lower than this.
  - V_{dup}: promote regions whose duplication ratios are higher than this.
  - V_{life}: regions living longer than this threshold can be effectively scanned.

- Default empirical values:
  - V_{cow} = 10%, V_{dup} = 20% and V_{life} = 100ms.

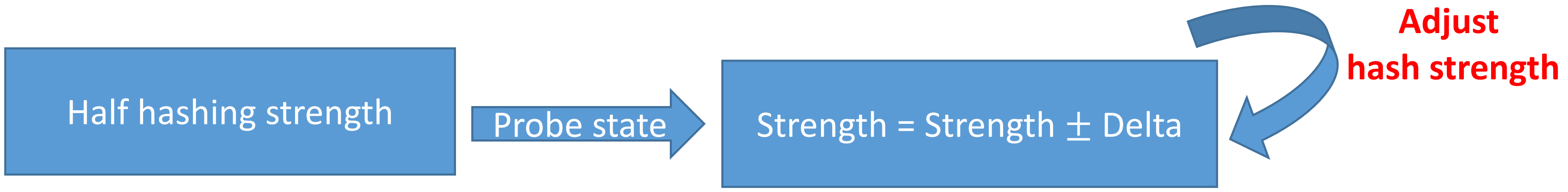
Various commercial products adopt UKSM and observe different sweet spots.

Hierarchical Region Distilling



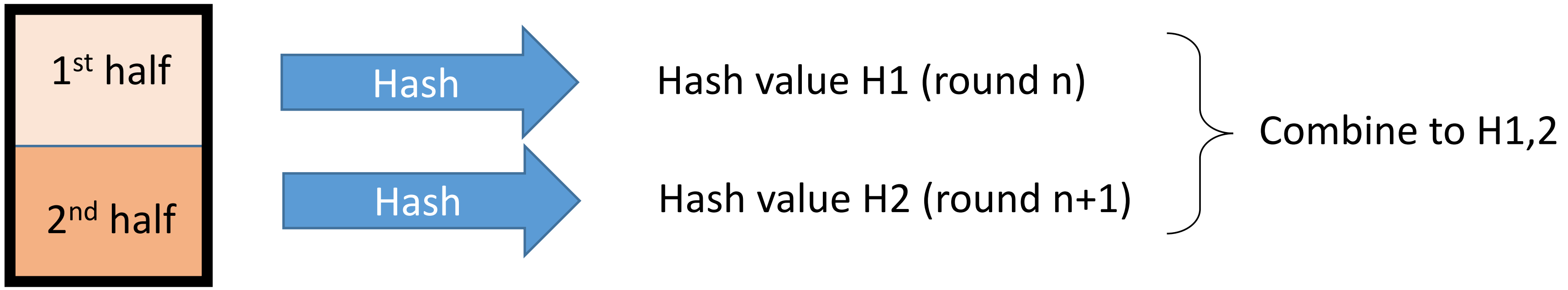
*: We adopt Linux KSM black-red tree design to track 'merged' and 'unmerged' pages.

Adaptive Partial Hashing



We optimize SuperFastHash with the following key contributions:

- Minimizing collisions – Optimizing **avalanche** for SuperFastHash [Hsieh 2004].
- Progressive hashing – Support **additivity** while adjust hash strengths.



Sampled page

Evaluation

- 6,000 Lines of Code in Linux kernel.
- OS: Vanilla kernel 4.4.
- Hardware:
  - Intel® Core ™ i7 CPU 920 with four 2.67 GHz cores.
  - 12 GB memory.
- For fair comparison
  - KSM is upgraded to SuperFastHash.

Evaluation Goals

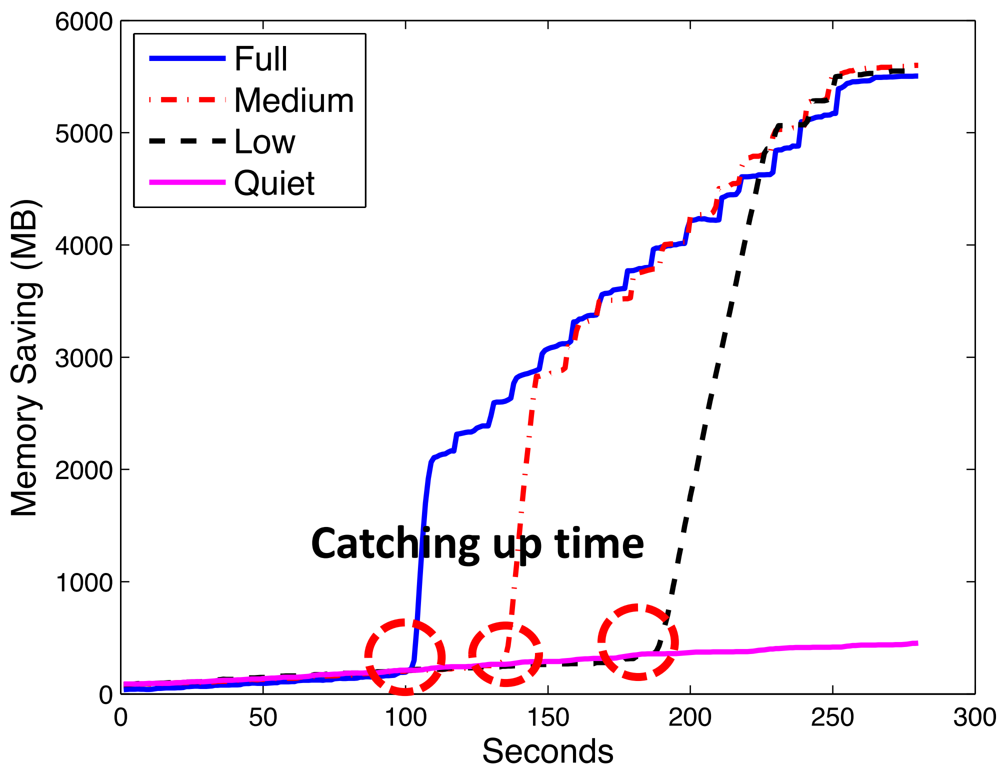
- How efficient is UKSM on different workloads?
- How flexible is UKSM regarding customization?
- What’s the responsiveness of UKSM vs KSM?
- How does adaptive partial hashing perform compared to non-adaptive algorithm?
- What’s the performance penalty of UKSM?

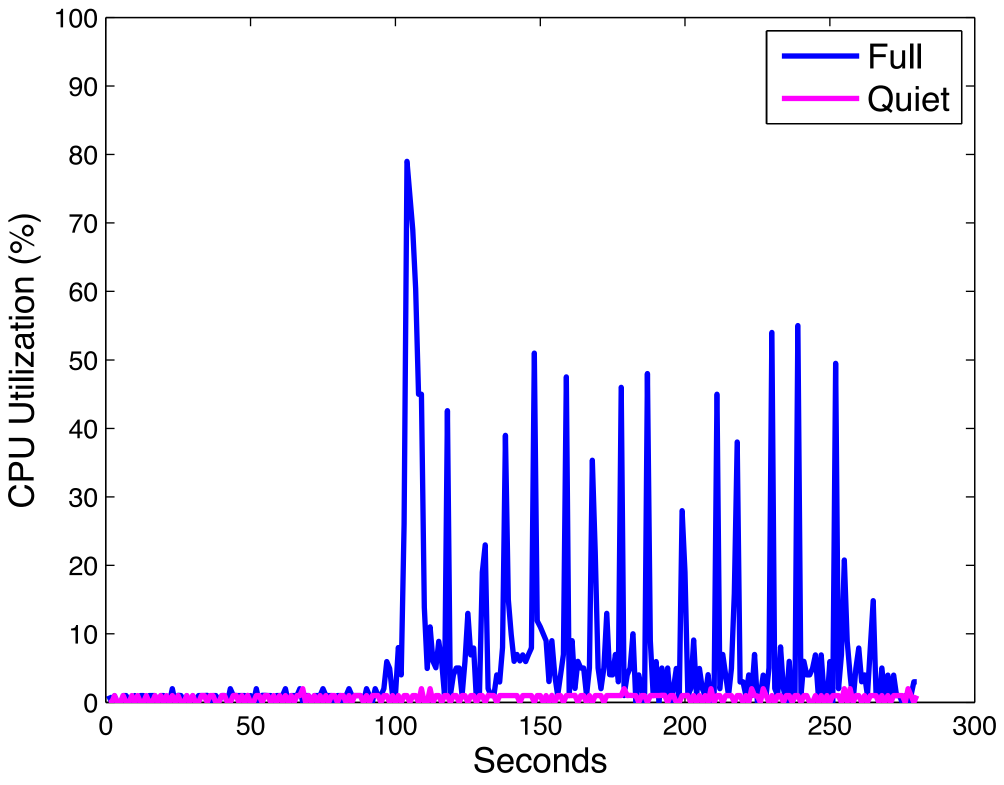
Evaluation Goals

- How efficient is UKSM on different workloads?
- How flexible is UKSM regarding customization?
- What’s the responsiveness of UKSM vs KSM?
- How does adaptive partial hashing perform compared to non-adaptive algorithm?
- What’s the performance penalty of UKSM?

Parameter Analysis

Setting: Booting 25 VMs, each with 1 VCPU, 1GB memory.





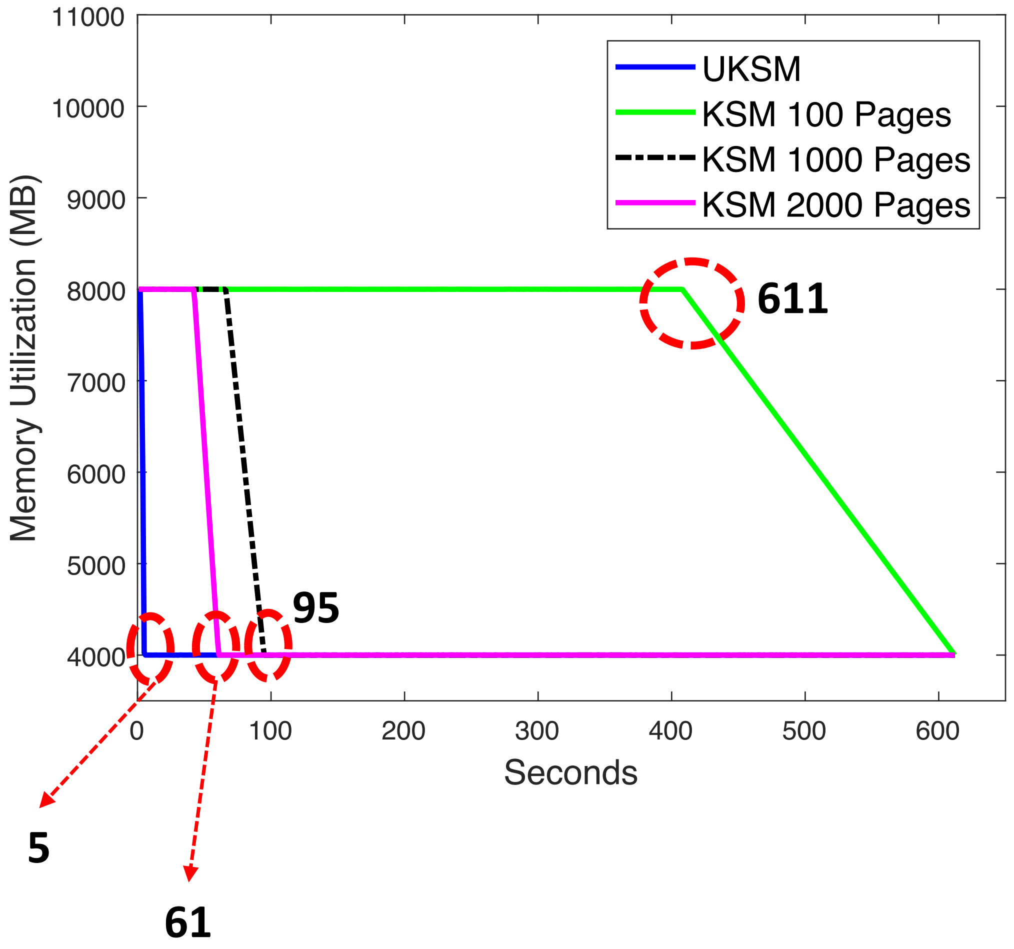
- UKSM allows four levels of scanning strengths:
  - Level **Full** allows upto 95% CPU consumption and can scan the entire memory in 2 seconds.
  - Each lower level will half the CPU and potentially increase the scan time by 2x.

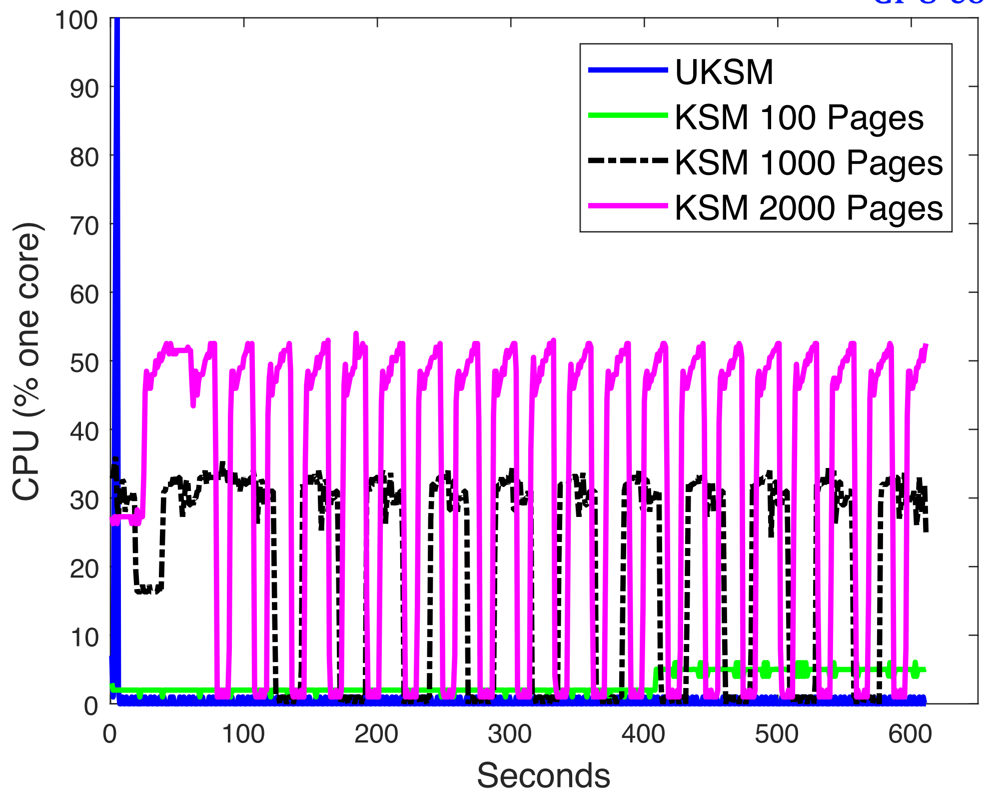
FAST'18

Responsiveness Analysis

Setting: Two processes, each with 4GB memory. One contains identical pages while the other random ones.

$$\text{Efficiency} = \frac{\text{memory saving}}{\text{CPU consumption}}$$





UKSM is 8.3×, 12.6×, 11.5× more efficient than KSM at scan speed of 100, 1000, 2000 pages.

Related Work

- Content-based approach:
  - VMware ESX server, IBM active memory deduplication, Red Hat ksmtuned.
  - Majority of them treat every page equally.

- I/O hint based approach:
  - KSM++ [Resolve 2012], XLH[Usenix ATC 2013], CMD [VEE 2014].
  - Cannot track anonymous memory space (no I/O) or require hardware change.

- SmartMD [Usenix ATC '17]:
  - Consider various page sizes; we are orthogonal.

UMASS LOWELL
Learning with Purpose

FAST '18

Conclusion

- Memory deduplication faces the unique challenges. Our techniques:
  - Hierarchical region distilling.
  - Adaptive partial hashing.

- UKSM saves 12.6x and 5x more memory than KSM on static and dynamic workload, respectively, in the same time envelope.

- UKSM is an in production system: https://github.com/dolohow/uksm.
- It has ~110 (watch, star and fork) after less than one year in GitHub.

UMASS LOWELL
Learning with Purpose

FAST'18

Thank You & Questions?

We would like to thank our shepherd Dr. Hong Jiang and anonymous reviewers!