Practical Online Cache Analysis and Optimization

Carl Waldspurger
Irfan Ahmad
CloudPhysics, Inc.

SNIA Legal Notice

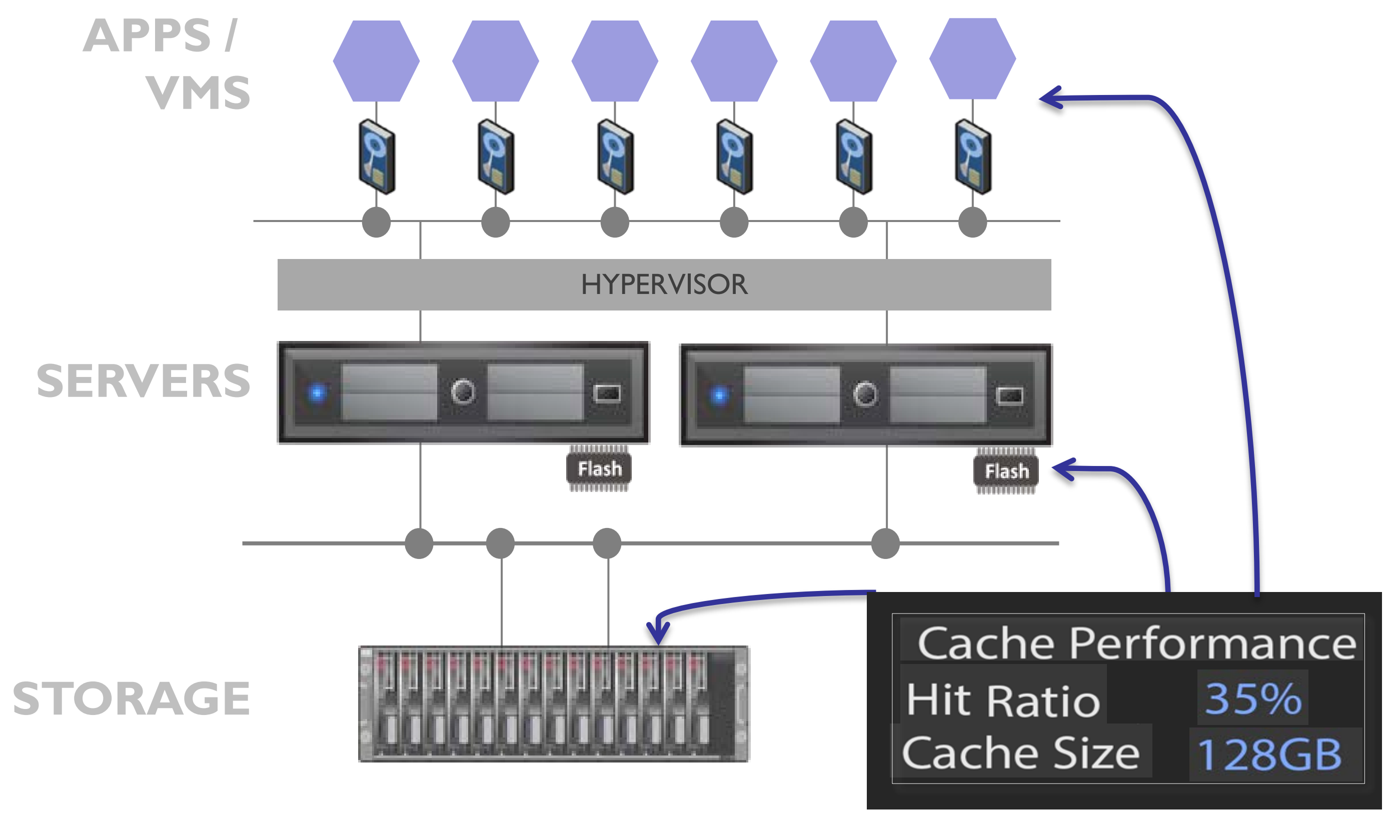
- The material contained in this tutorial is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations.
- This presentation is a project of the SNIA Education Committee.
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be, or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved. The author, the presenter, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

Abstract

- Practical Online Cache Analysis and Optimization
  - The benefits of storage caches are notoriously difficult to model and control, varying widely by workload, and exhibiting complex, nonlinear behaviors. However, recent advances make it possible to analyze and optimize high-performance storage caches using lightweight, continuously-updated miss ratio curves (MRCs). Previously relegated to offline modeling, MRCs can now be computed so inexpensively that they are practical for dynamic, online cache management, even in the most demanding environments. We show how MRCs can be leveraged to guide efficient cache sizing, allocation, and partitioning, supporting diverse goals such as improved performance, isolation, and quality of service. We will also describe how multiple MRCs can be used for online tuning of cache parameters and policies.

Caches Pervasive in Modern Storage



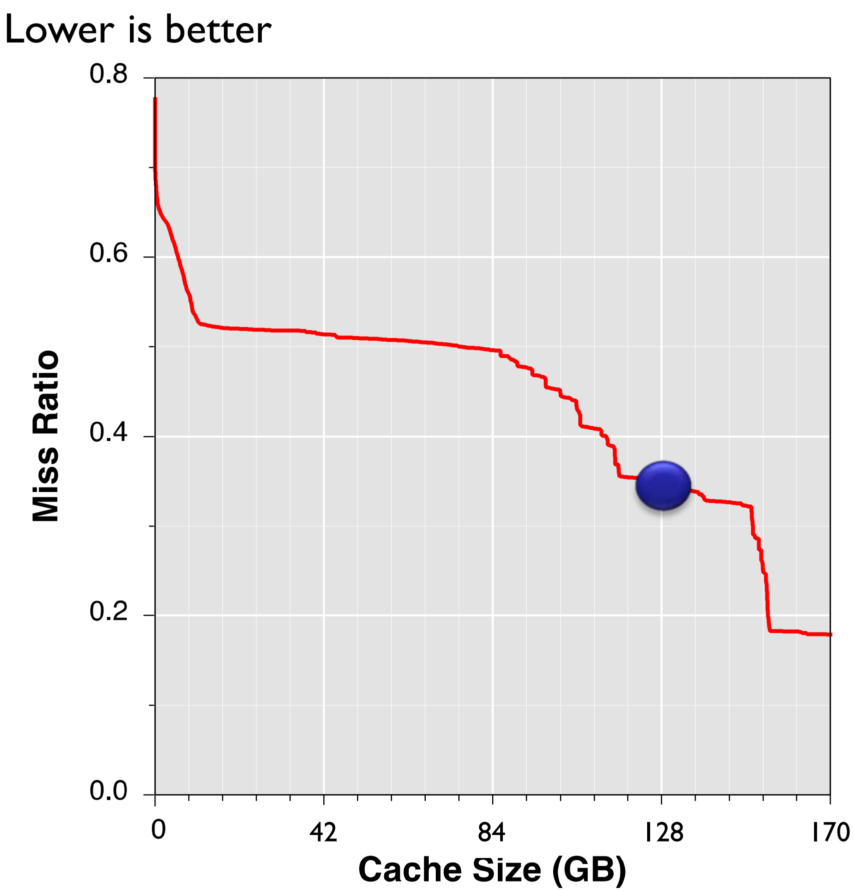
Cache Performance Questions

- Is this performance good? Can it be improved?
- What happens if I add / remove some cache?
- What if I add / remove workloads?
- Is there cache thrashing / pollution?
- What if I change cache algorithm parameters?

Problem & Opportunity

- Cache performance highly non-linear
- Benefit varies widely by workload
- Opportunity: dynamic cache management
  - Efficient sizing, allocation, and scheduling
  - Improve performance, isolation, QoS
- Problem: online modeling expensive
  - Too resource-intensive to be broadly practical
  - Exacerbated by increasing cache sizes

Modeling Cache Performance



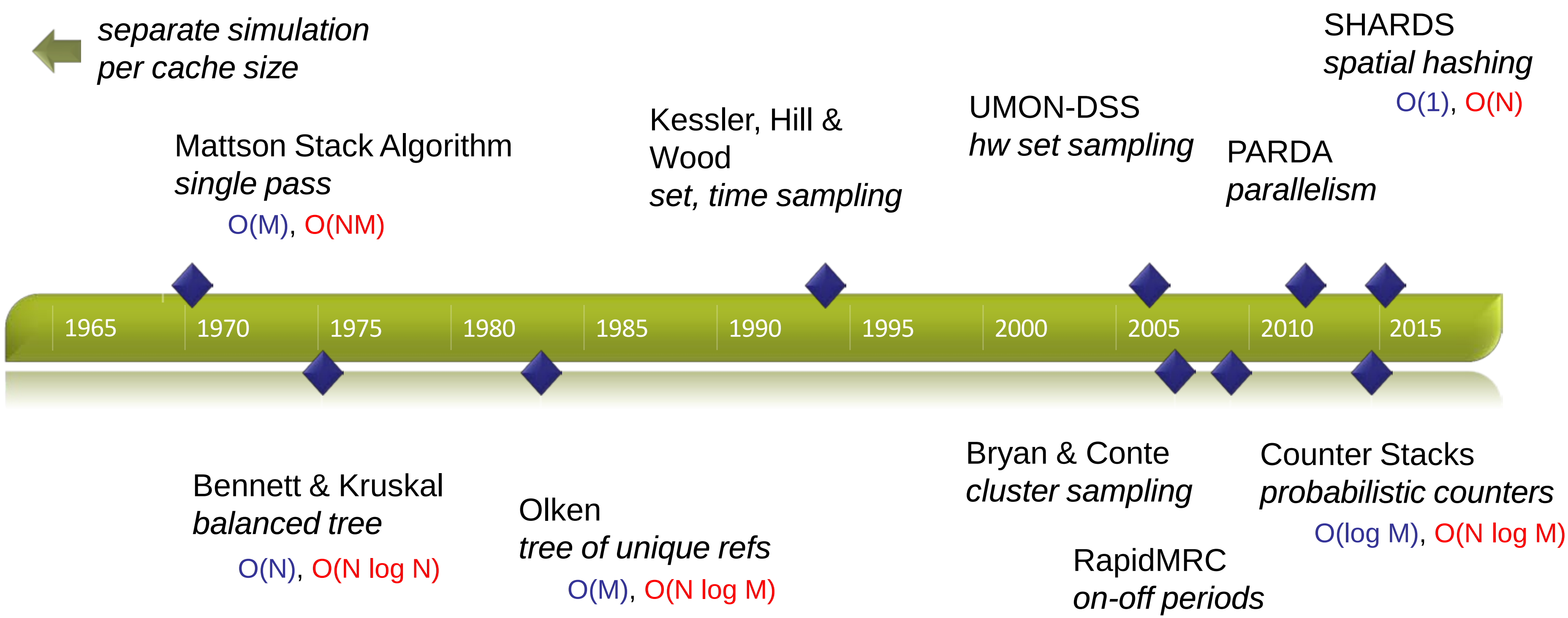
- Miss Ratio Curve (MRC)
  - Performance as *f*(size)
  - Working set knees
  - Inform allocation policy
- Reuse distance
  - Unique intervening blocks between use and reuse
  - LRU, stack algorithms

Mattson Algorithm Example

			✗	✗	✓	✓	✓	
references	…	C	B	A	D	A	B	C
distances	…	4	∞	3	7	1	2	3

- Classic single-pass method (IBM 1970)
- Reuse distance
  - Unique references since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

MRC Algorithm Research



Space, Time Complexity

N = total refs, M = unique refs

New Advances: MRC Approximations

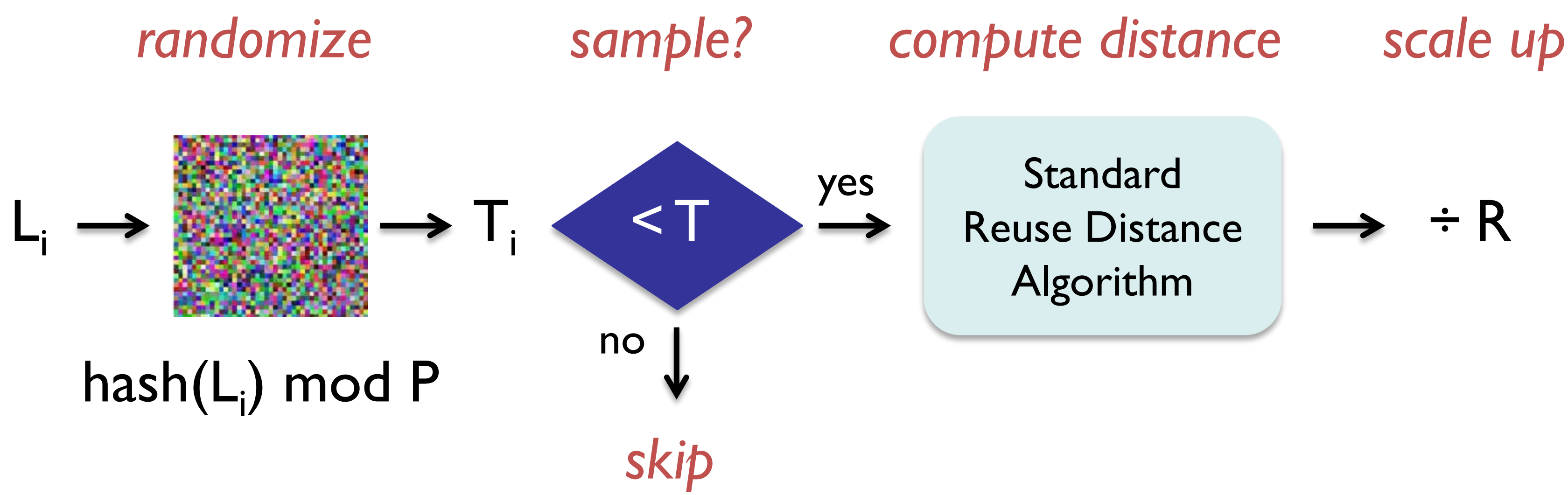
Counter Stacks (2014)

- Efficient approx counting, downsampling, pruning
- Uses probabilistic counters to track block reuse
- Supports checkpoints with splicing/merging
- High performance in small O(lg M) memory footprint
- Highly accurate MRCs
- LRU (stack algorithms)

SHARDS (2015)

- Efficient randomized spatial sampling
- Runs full MRC algorithm, using only sampled blocks
- Hashing to capture all reuses of same block
- High performance in tiny O(1) constant footprint
- Highly accurate MRCs
- Generalizes to non-LRU

Closer Look: SHARDS



Sampling rate R = T / P

Each sample statistically represents 1/R blocks

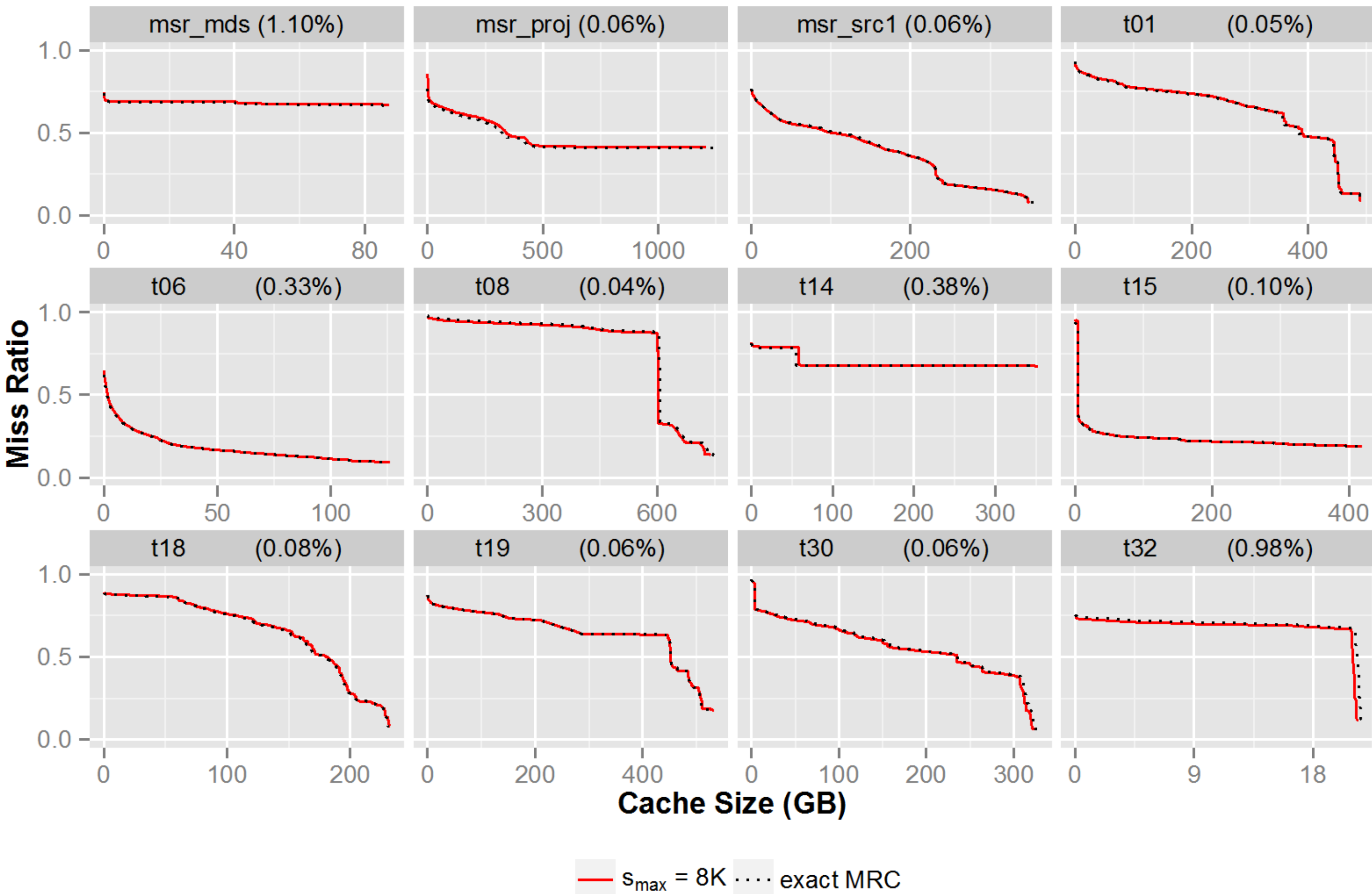
Bound sample set by lowering T dynamically

Example Systems Implementation

- Easy integration with existing embedded systems
- Example C interface
  - `void mrc_process_ref(MRC *mrc, LBN block);`
  - `void mrc_get_histo(MRC *mrc, Histo *histo);`
- Extremely low resource usage (SHARDS)
  - Accurate MRCs in <1 MB footprint
  - Single-threaded throughput of 17-20M blocks/sec
  - Average time of mrc_process_ref() call < 60 ns
  - No floating-point, no dynamic memory allocation
- Scaled-down simulation similarly efficient

MRC Accuracy (LRU, Real Workloads)

Lower is better



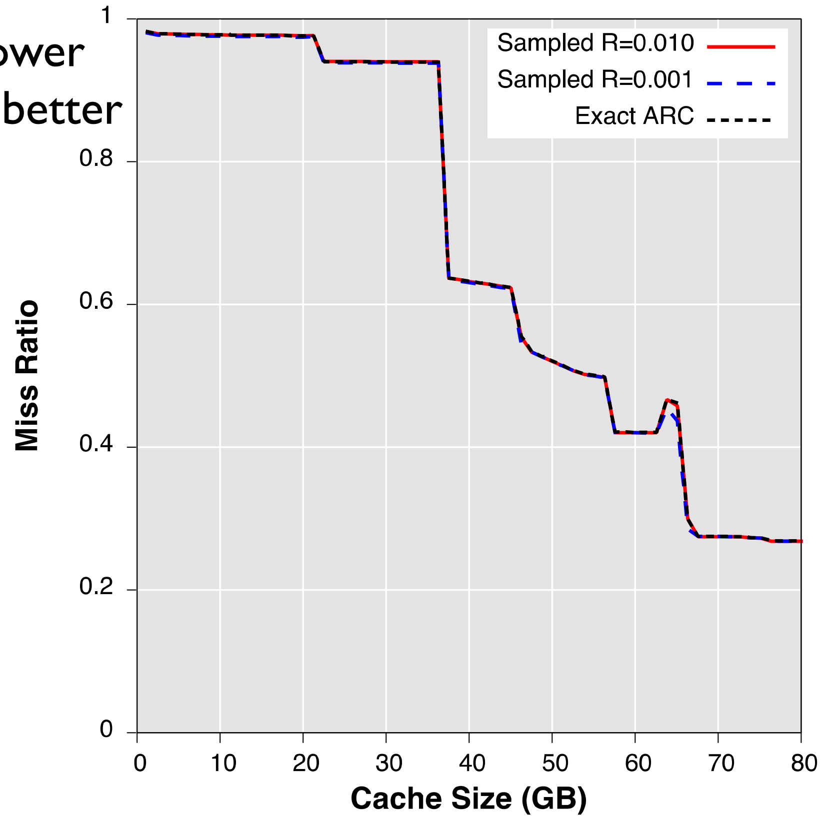
Generalizing to Non-LRU Policies

- Sophisticated caching algorithms
  - ARC, LIRS, CAR, Clock-Pro, 2Q, …
  - No known single-pass methods!
- Scaled-down simulation
  - Leverages SHARDS hashed spatial sampling
  - Simulate each size separately
- Still highly efficient
  - Low sampling rate R = 0.001
  - 1000× reduction in memory, processing
  - 100× for concurrent simulation of 10 cache sizes!

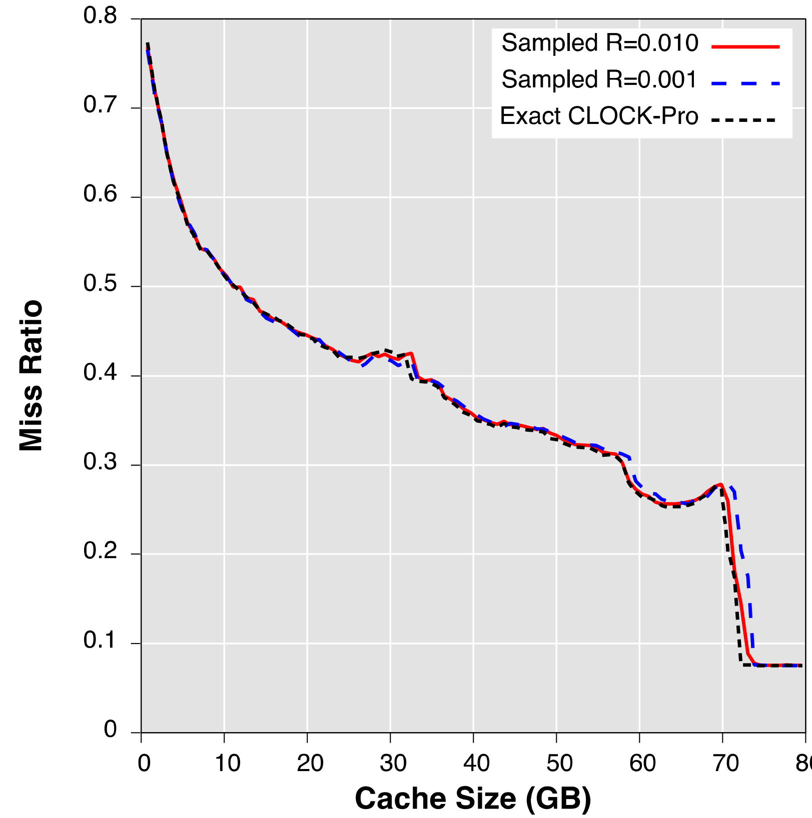
Non-LRU MRC Accuracy

Lower is better

ARC — MSR-Web Trace

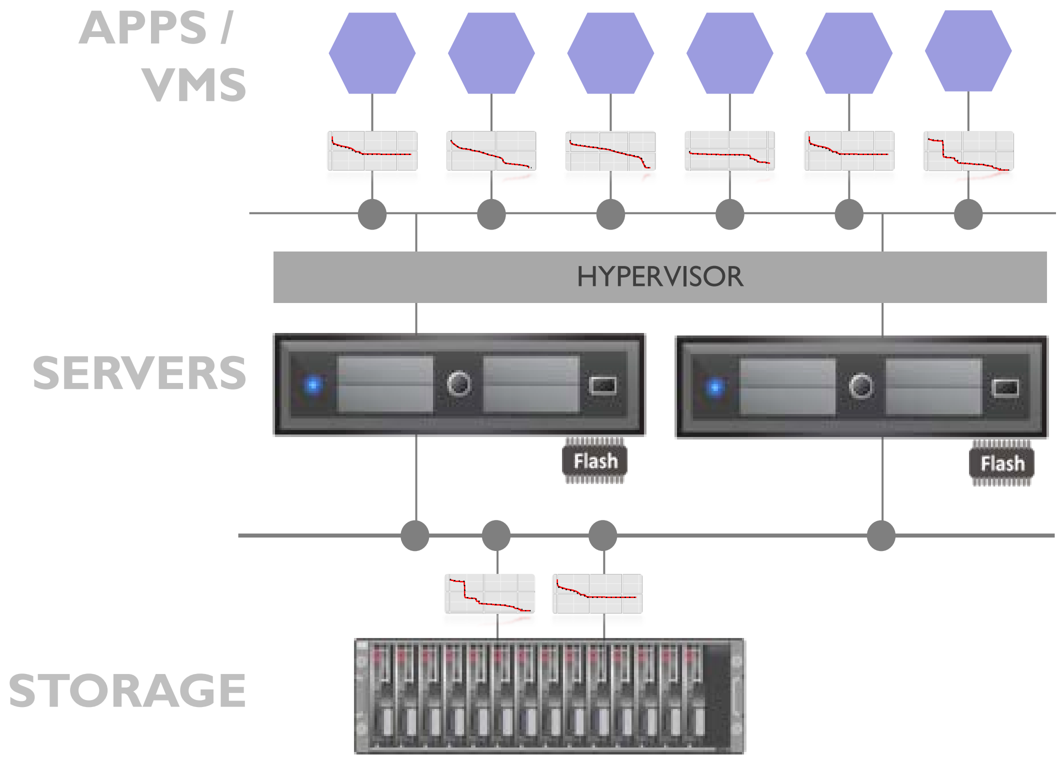


CLOCK-Pro — Trace *t04*



Applications of Online MRCs

Where are the MRCs?



Overview of Use Cases

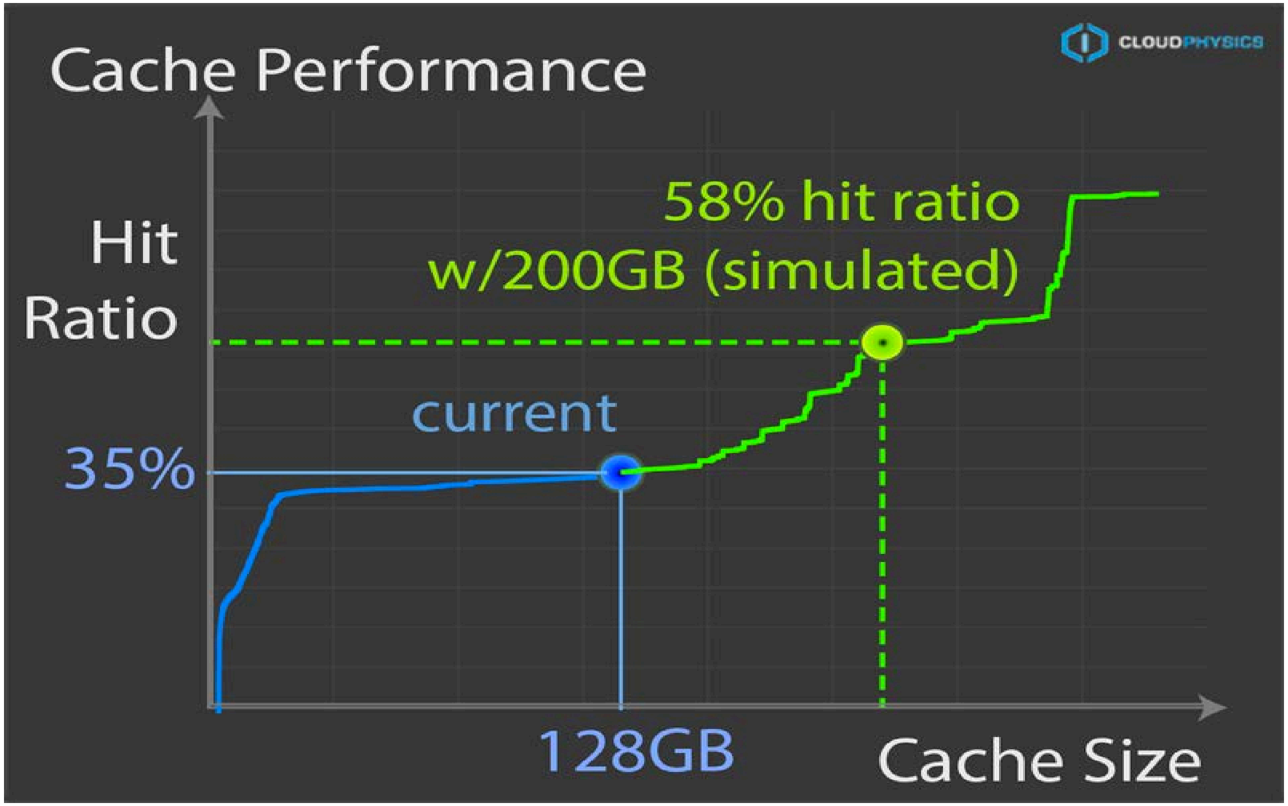
- Without any changes to cache
  - Cache sizing
  - Cache parameter tuning
- With cache partitioning support
  - Optimize aggregate performance
  - Isolate individual clients
- Guarantee SLAs
  - Latency
  - Throughput

Use Case: Cache Sizing

- Online recommendations
  - Integrate MRCs with storage controller
  - Tune and optimize customer workloads
- Show MRCs in storage management UI
  - Report cache size to achieve desired latency
  - Customers and SEs self-service on sizing
  - Size array cache in the field, trigger upsell, etc.

Example Cache Sizing UI (Mockup)

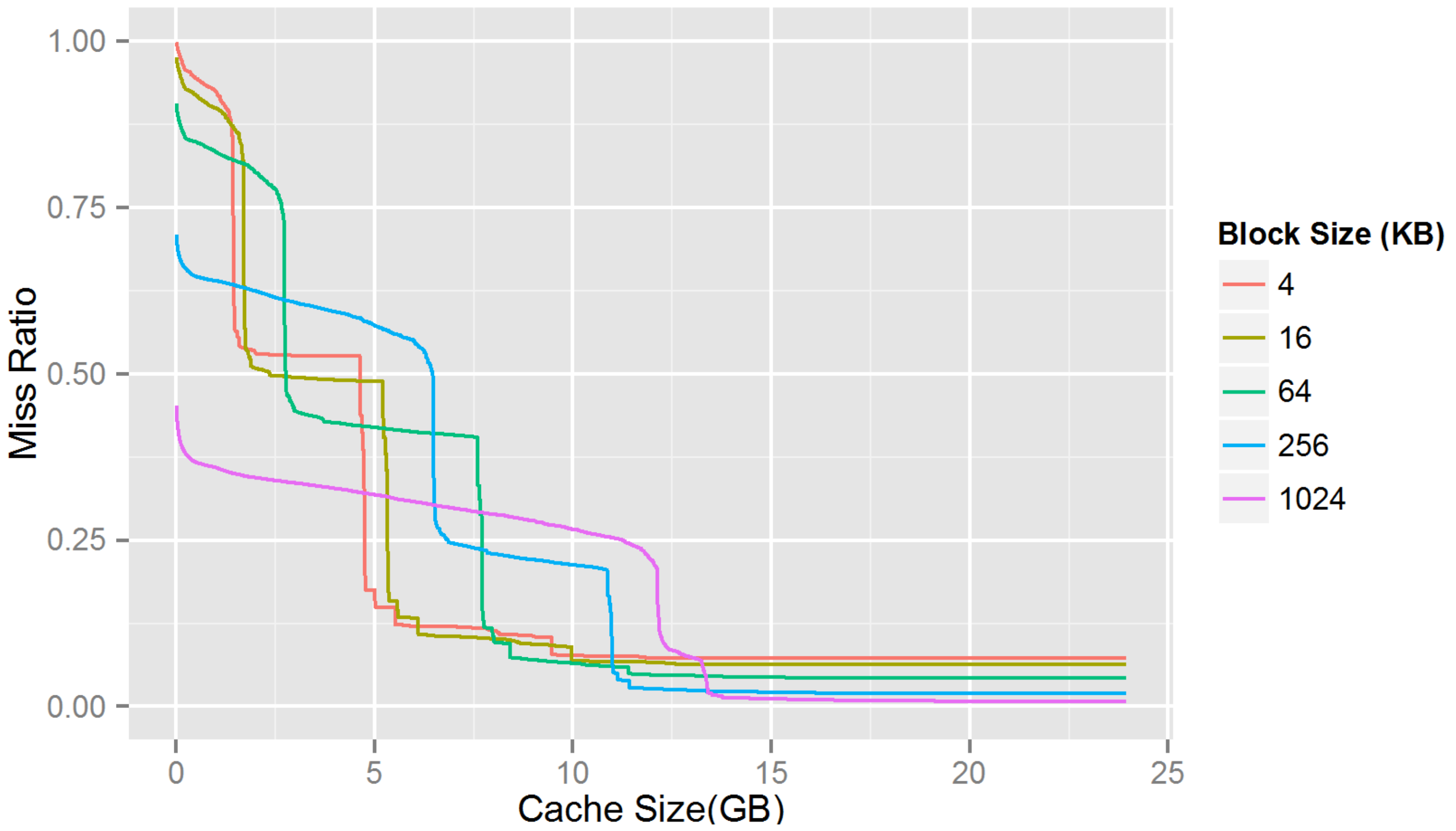
Higher
is better



Use Case: Tune Cache Policy

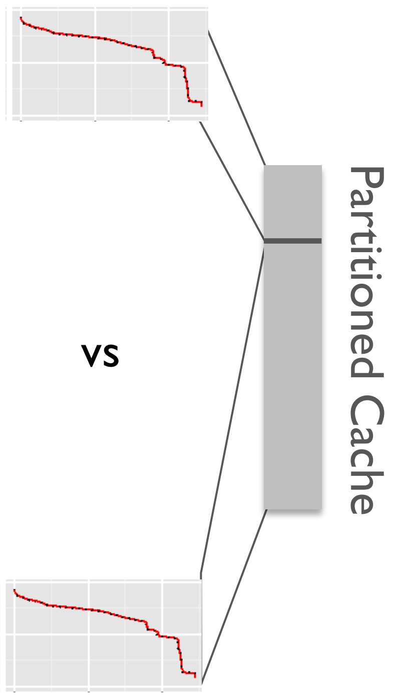
- Quantify impact of parameter changes
  - Cache block size, use of sub-blocks
  - Write-through vs. write-back
  - Even replacement policy…
- Explore without modifying *actual* production cache
  - Simulate multiple configurations concurrently
  - Multiple MRCs, each with different parameters
- Dynamic online optimization
  - Determine best configuration
  - Adjust actual cache parameters

Example: Cache Block Size Tuning

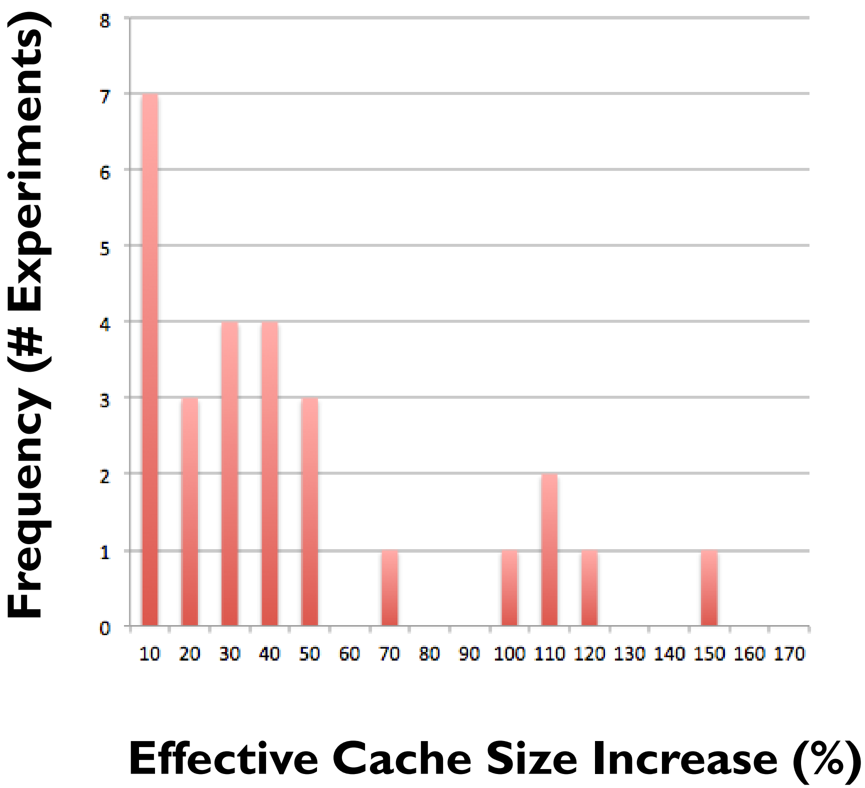


Use Case: Optimize Performance

- Improve aggregate cache performance
  - Allocate space based on client *benefit*
  - Prevent inefficient space utilization
- Mechanism: Partition cache across clients
  - Isolate and control competing LUNs, VMs, tenants, DB tables, etc.
  - Optimize partition sizes using MRCs
- Adapt to changing workload behavior



Example: Partitioning Results

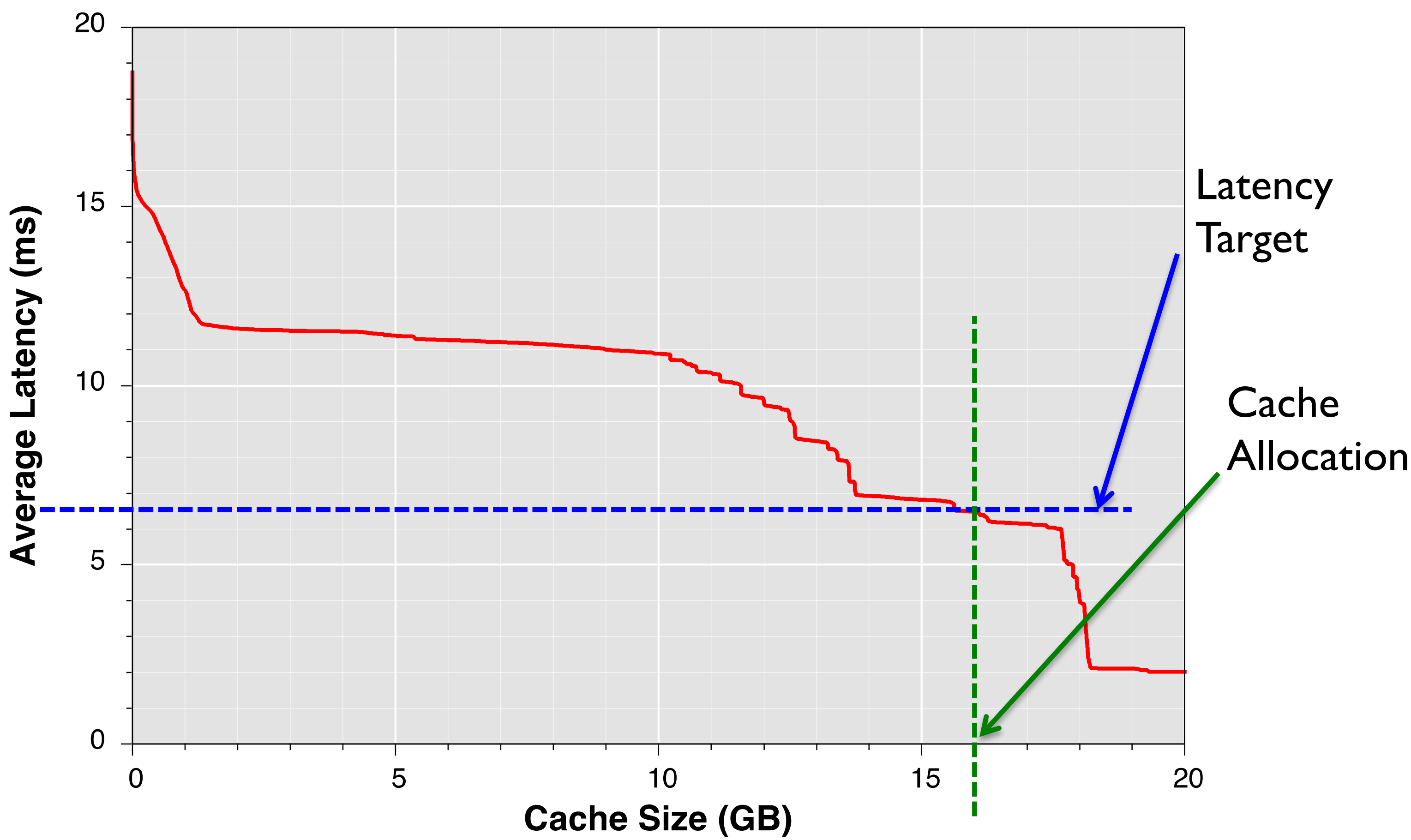


- Customer traces
  - 27 workload mixes
  - 8, 32, 128 GB cache sizes
- SHARDS partitions vs. global LRU
- Results histogram
  - Effective cache size
  - 40% larger (avg)
  - 146% larger (max)

Use Case: Latency, IOPS Guarantees

- Meet service-level objectives
  - Per-client latency or throughput targets
  - Use cache allocation as general QoS knob
- Same partitioning mechanism
  - Isolate and control LUNs, VMs, DB tables, tenants, etc.
  - Use MRCs for sizing partitions to meet goals
- Adapt to changing workload behavior

Example: Achieving Latency Target



Conclusion

- Miss Ratio Curves (MRCs)
  - Powerful, game-changing storage tool
  - New algorithms use dramatically less resources
- Online MRCs now practical
  (data from CloudPhysics licensable implementation)
  - ~20 million IO/s per core; amortized 60 ns per IO
  - Extremely high accuracy in 1 MB footprint
  - Feasible for for memory-constrained firmware, drivers
- Compelling use cases
  - Workload-aware predictive cache sizing and tuning
  - Software-driven cache partitioning for “free” performance
  - Latency / throughput guarantees via cache QoS

Attribution & Feedback

The SNIA Education Committee thanks the following Individuals for their contributions to this Tutorial.

Authorship History

23-Nov-15 Carl Waldspurger, Irfan Ahmad

Additional Contributors

Please send any questions or comments regarding this SNIA Tutorial to ***tracktutorials@snia.org***