SDC

STORAGE DEVELOPER CONFERENCE

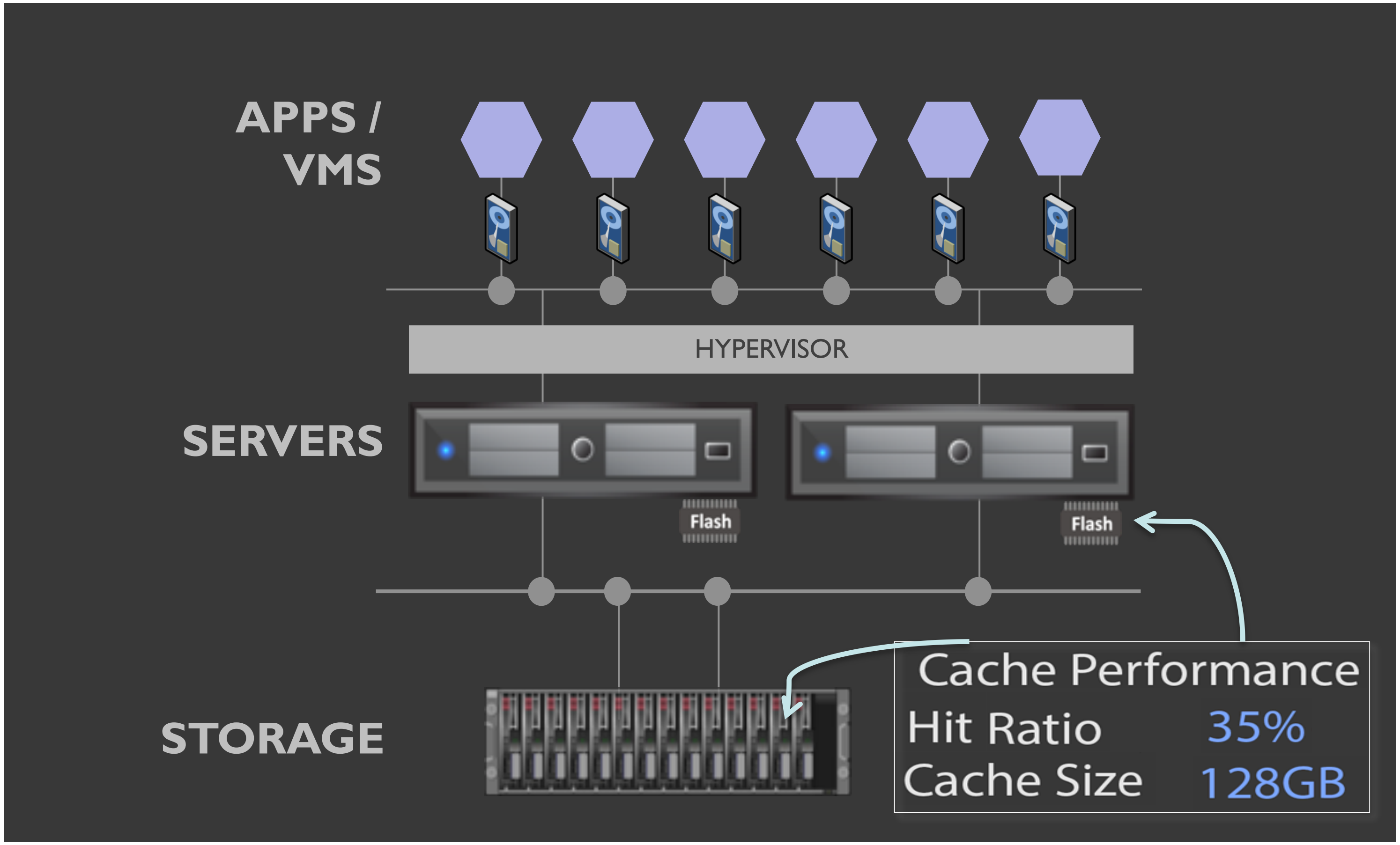
SNIA ■ SANTA CLARA, 2015

Online Cache Analysis And Its Applications For Enterprise Storage Systems

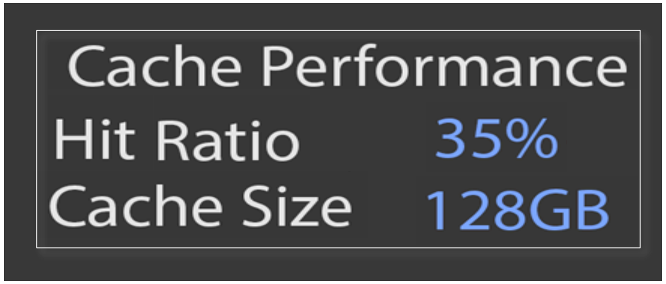
Irfan Ahmad

Founder & CTO
CloudPhysics, Inc.

System Model – Caches are Critical



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
- Can I use cache to control performance or QoS

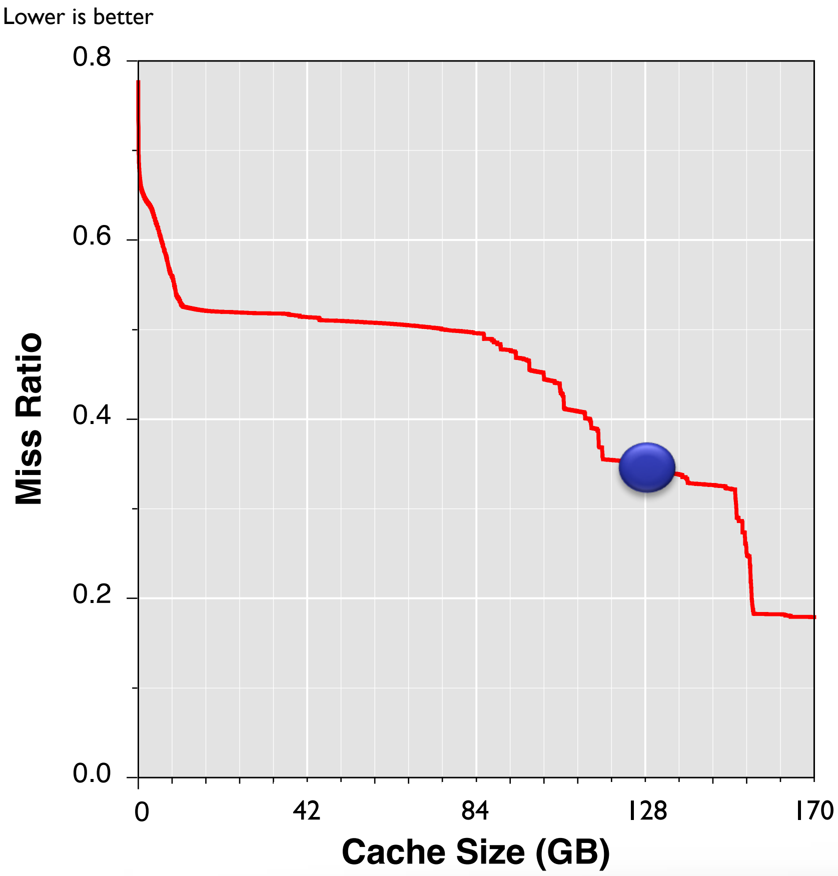
Executive Summary

- Miss Ratio Curves (MRCs): game-changing storage tool
- CloudPhysics' MRC algorithm = up to 10,000x improvement
- Online MRCs now practical:
  - ~20 million IO/s per core; amortized 60 ns per IO
  - High accuracy in 1 MB
  - Feasible for for memory-constrained firmware, drivers
- Looking to partner with storage systems vendors
- Applications:
  - Workload-aware predictive cache sizing
  - Software-driven cache partitioning for "free" performance
  - Latency / Throughput guarantees via cache QoS

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

MRC Algorithm Research

separate simulation per cache size

Mattson Stack Algorithm
single pass
O(M), O(NM)

Kessler, Hill & Wood
set, time sampling

UMON-DSS
hw set sampling

PARDA
parallelism

SHARDS
spatial hashing
O(1), O(N)

Bennett & Kruskal
balanced tree
O(N), O(N log N)

Olken
tree of unique refs
O(M), O(N log M)

Bryan & Conte
cluster sampling

RapidMRC
on-off periods

Counter Stacks
probabilistic counters
O(log M), O(N log M)

Space, Time Complexity
N = total refs, M = unique refs

SDC 15

Key New Idea

- CloudPhysics MRC approximation algorithm
  - Randomized spatial sampling
  - Hashing to capture all reuses of same block
  - High performance in tiny constant footprint
  - Highly accurate MRCs
- Summary: run *full* algorithm, using *sampled* blocks
  - https://www.usenix.org/conference/fast15/technical-sessions/presentation/waldspurger

Licensable Systems Implementation

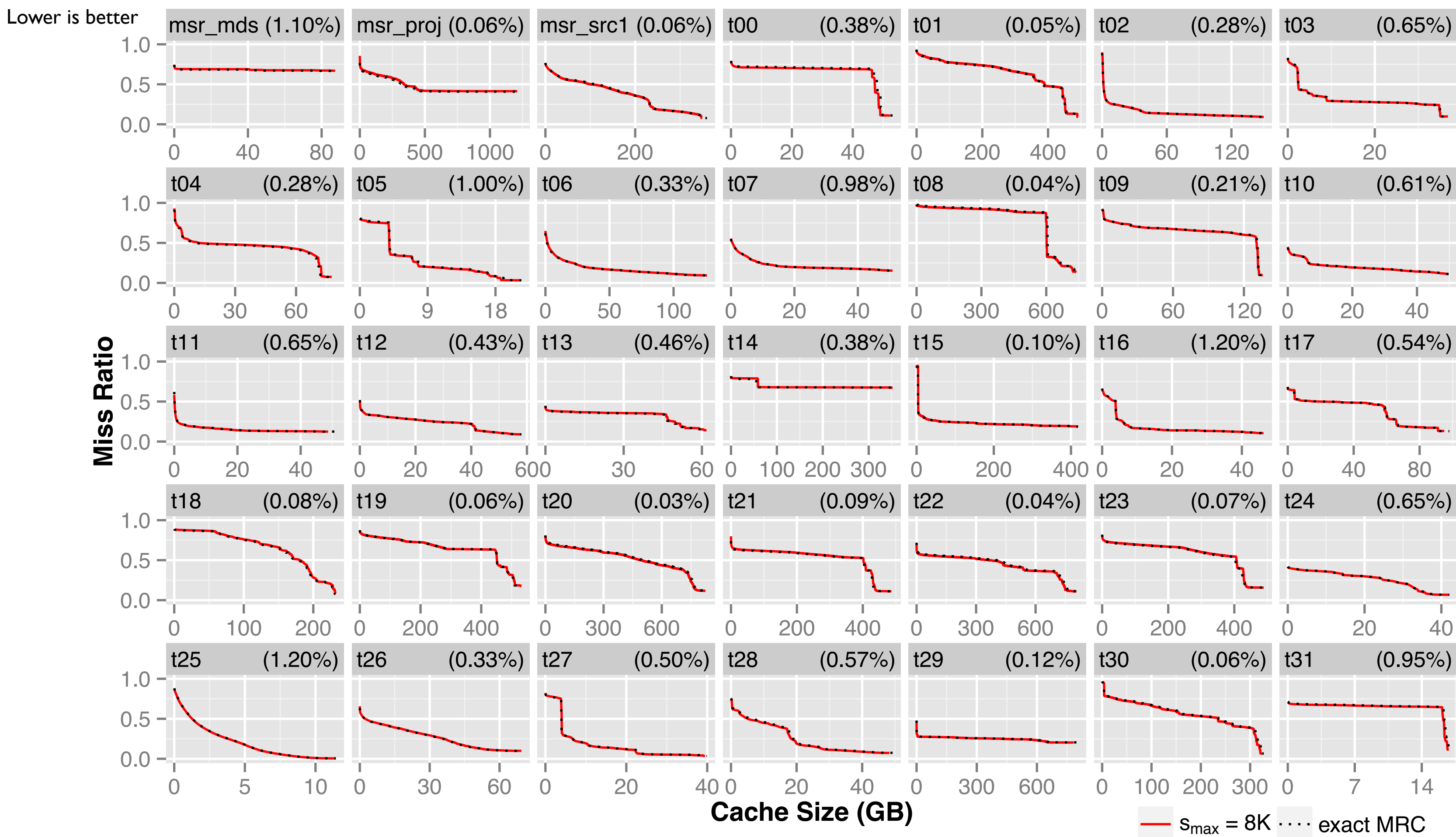
- Easy to integrate with existing embedded systems
  - High-performance SHARDS implementation in C

```
void mrc_process_ref(MRC *mrc, LBN block);
void mrc_get_histo(MRC *mrc, Histo *histo);
```

  - No floating-point, no dynamic memory allocation
- Extremely low resource usage
  - Accurate MRCs in <1 MB footprint
  - Single-threaded throughput of 17-20M blocks/sec
  - Average time of `mrc_process_ref()` call < 60 ns
- Scaled-down simulation similarly efficient

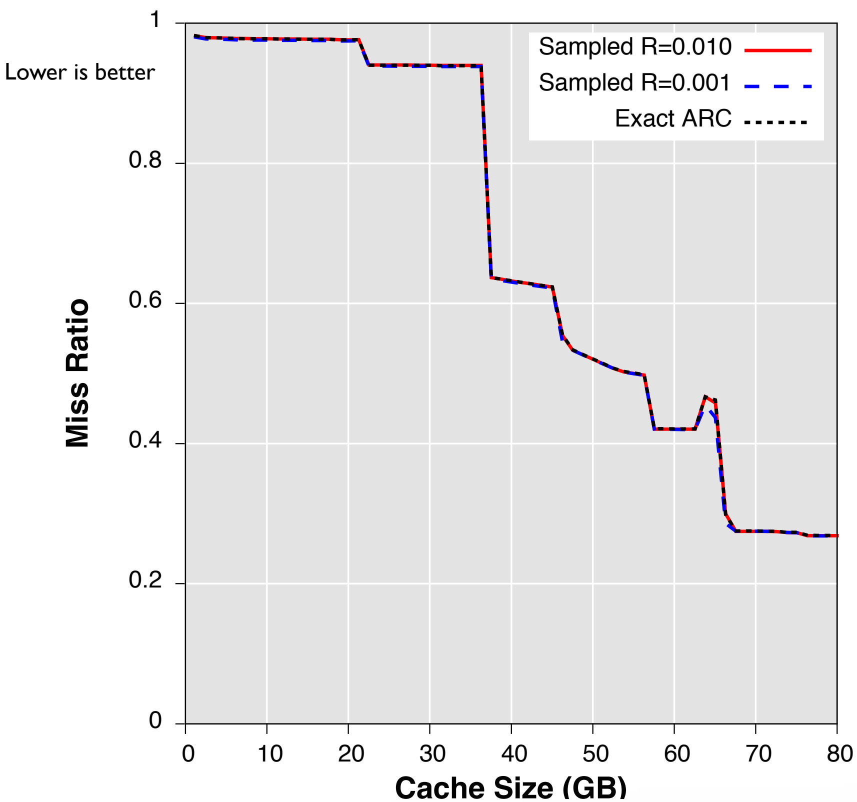
SDC 15

MRCs from Customer Workloads (LRU)

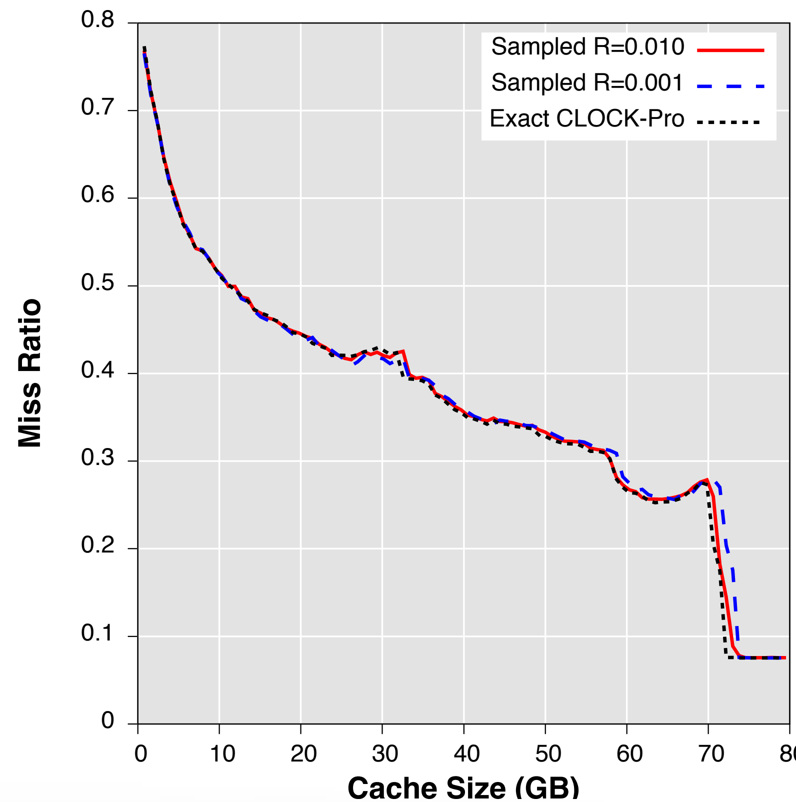


Non-LRU Miss Ratio Curve Examples

ARC — MSR-Web Trace

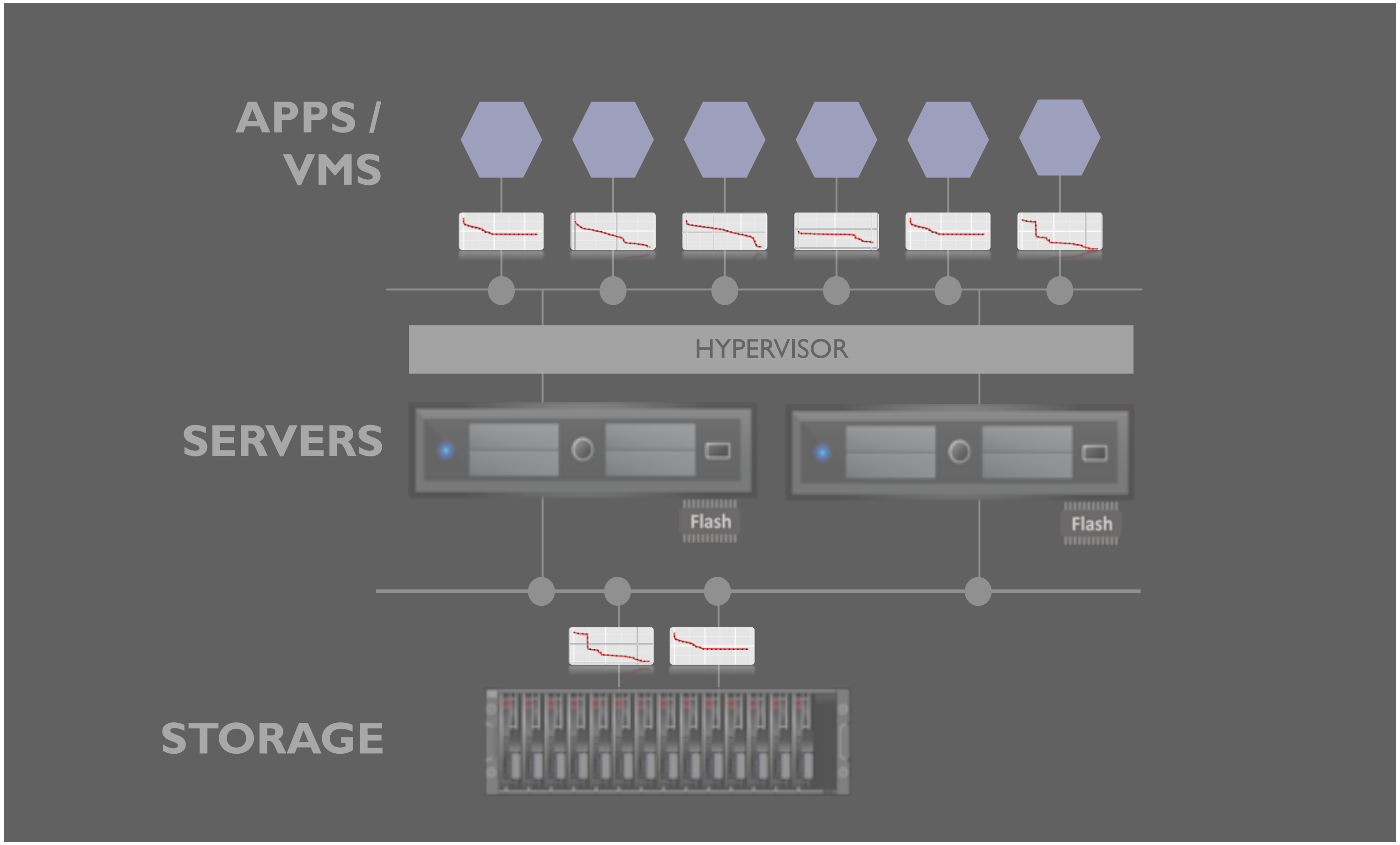


CLOCK-Pro — Trace *t04*



Applications of Online MRC

Where are the MRCs



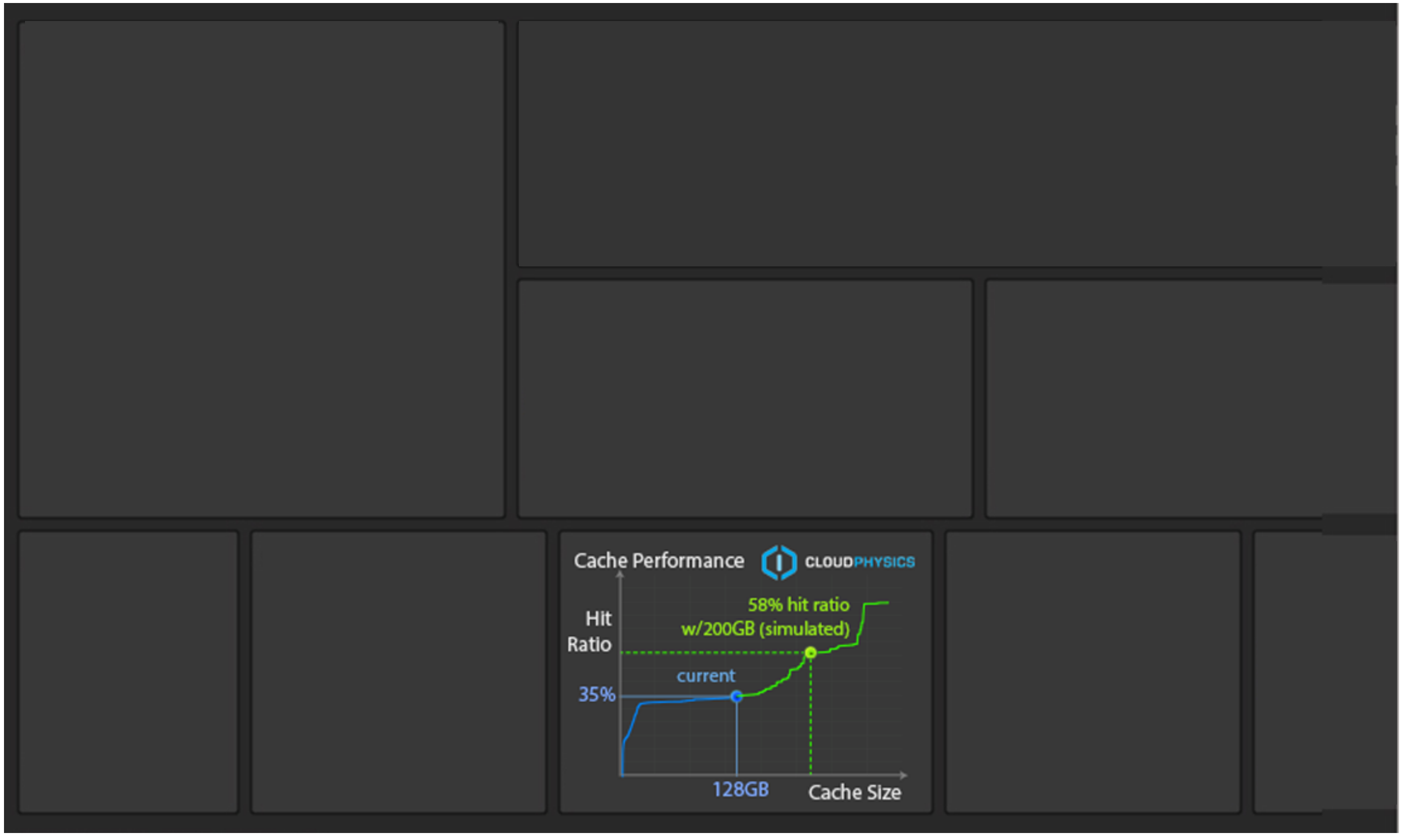
Overview of Applications

- Without any changes to cache
  - Cache sizing
  - Cache parameter tuning
- With cache partitioning support
  - Optimize performance
  - Enforce service-level objectives
- Next-generation optimizations
  - Latency and throughput guarantee SLAs
  - AI for bending MRC curves

Application: Cache Sizing

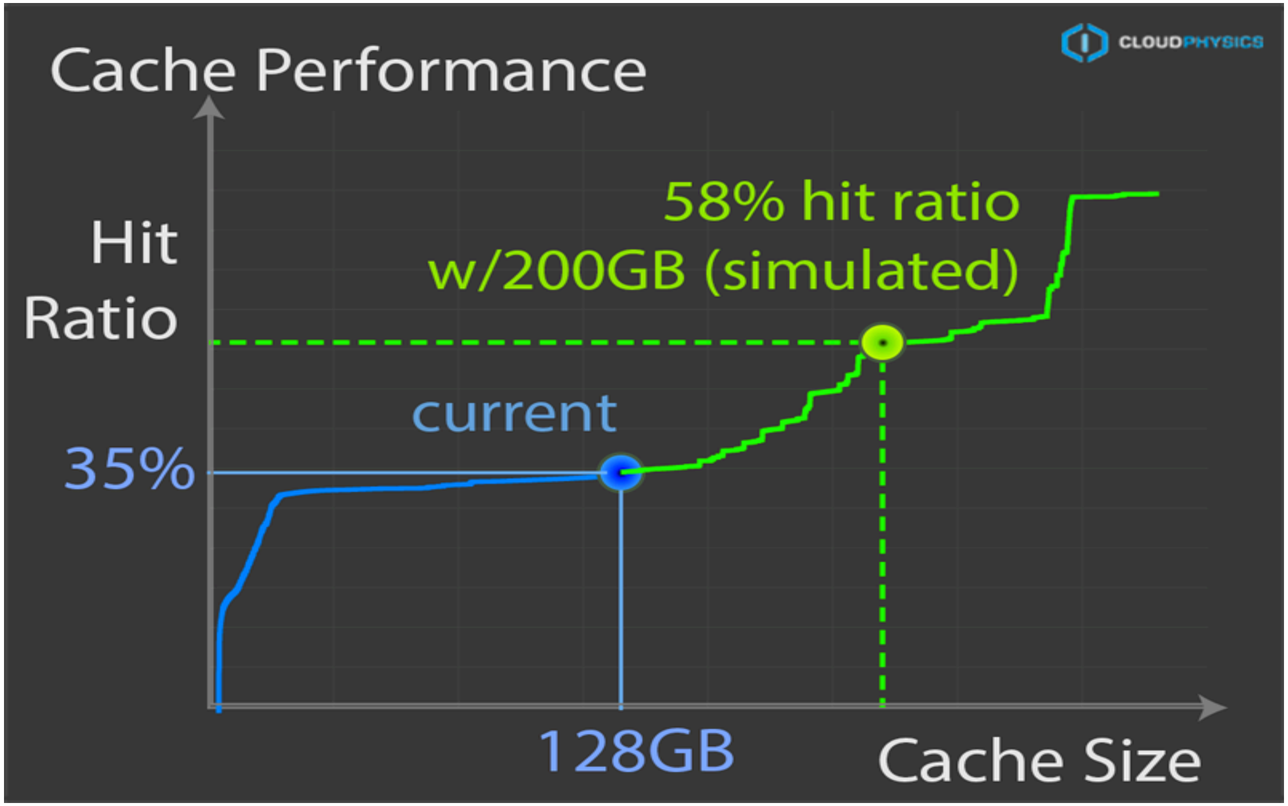
- Online recommendations
  - Integrate SHARDS with storage controller
  - Show MRCs in storage management UI
  - Customers and SEs self-service on sizing
  - Size array cache in the field, trigger upsell, etc.
  - Tune and optimize customer workloads
  - Report cache size to achieve desired latency
-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

Example: MRC in UI Dashboard (mockup)



Example: MRC UI Mockup

Higher is better

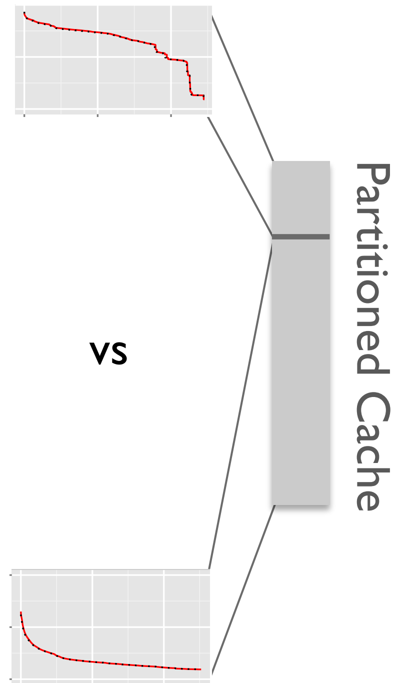


Application: Tune Cache Policy

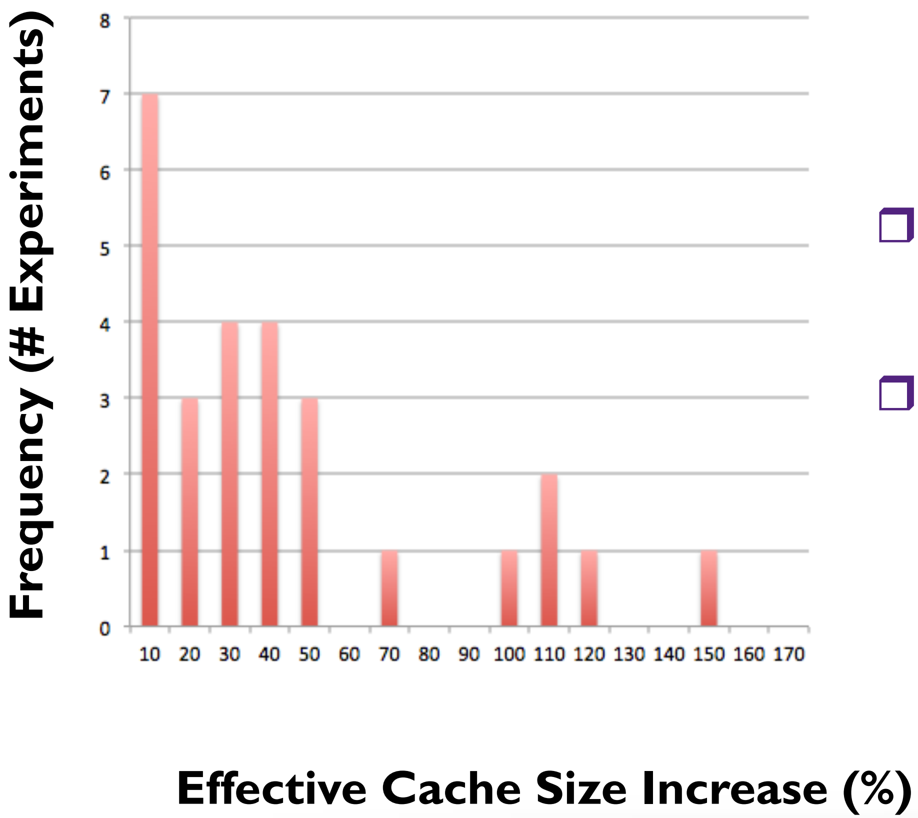
- Quantify impact of parameter changes
  - Cache block size, use of sub-blocks
  - Write-through vs. write-back
  - Even replacement policy…
- Scaled-down simulation
  - Representative “microcosm” of cache behavior
  - Works for arbitrary policies and parameters
  - Explore without modifying *actual* production cache
- Dynamic online optimization

Application: Optimize Performance

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



Example Partitioning Results

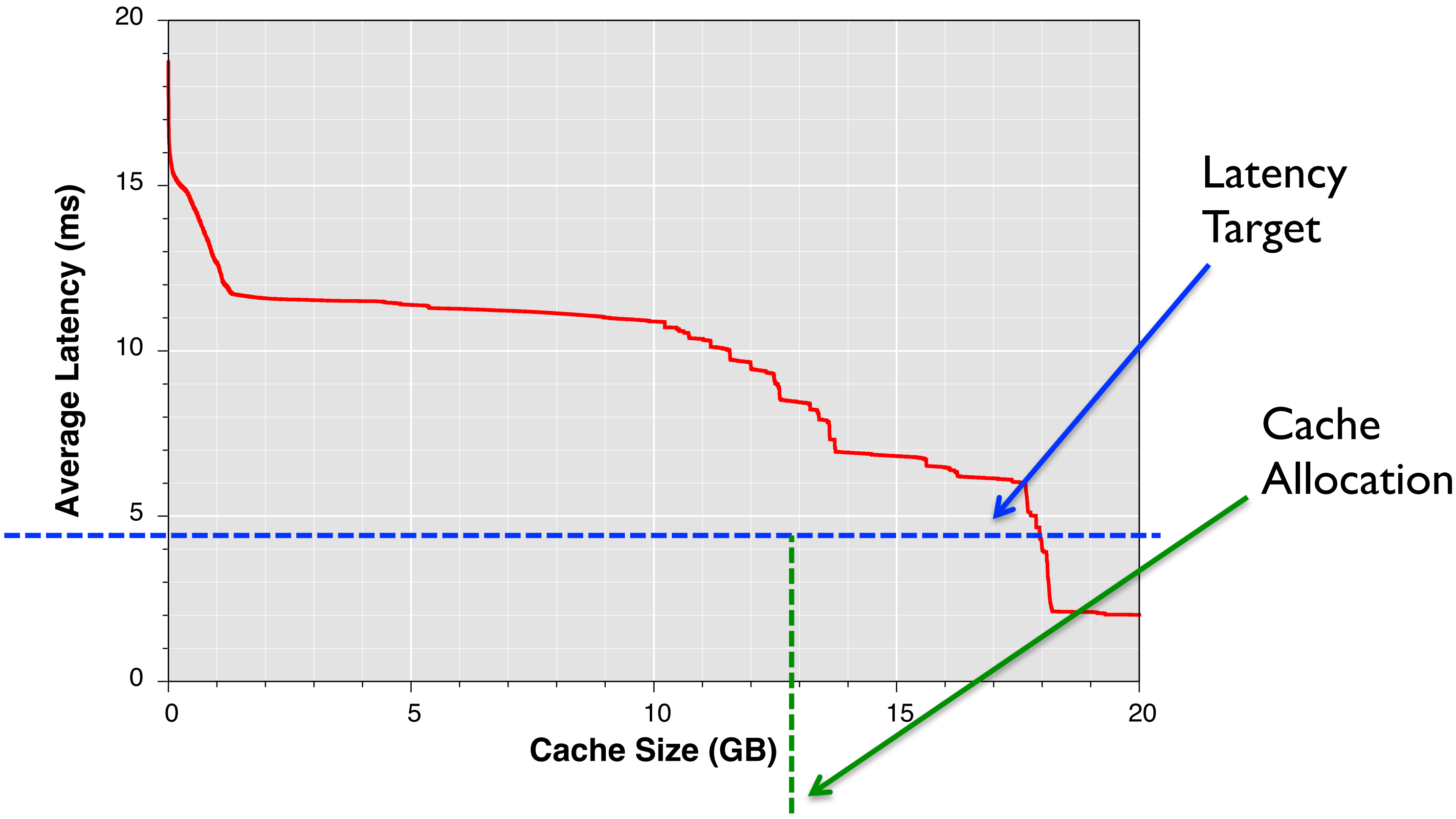


- Customer traces
  - 27 workload mixes
  - 8, 32, 128 GB sizes
- SHARDS partitions vs. global LRU
- Results histogram
  - Effective cache size
  - 40% larger (avg)
  - 146% larger (max)

Application: 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, tenants, etc.
  - Use MRCs for sizing partitions to meet goals
- Adapt to changing workload behavior

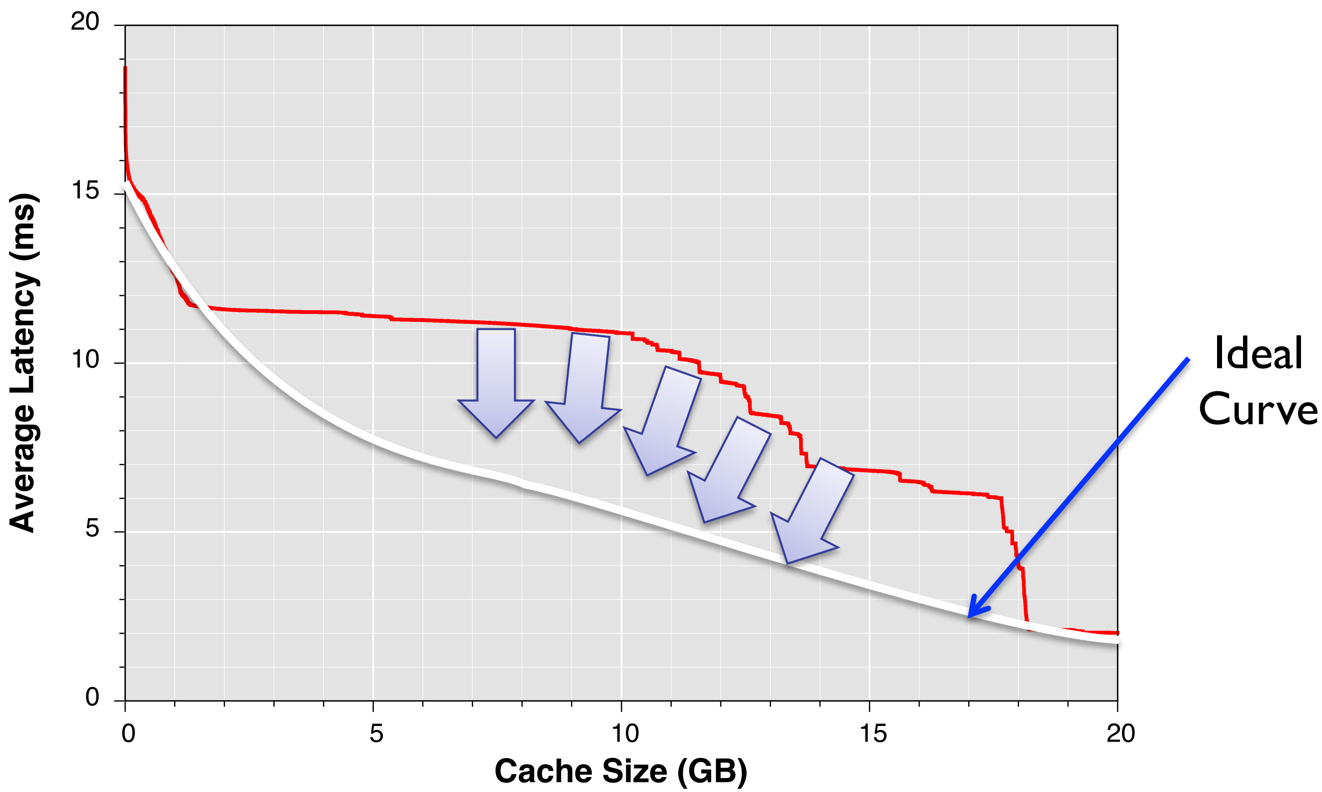
Example: Achieving Latency Target



Next-Generation Capabilities

- Unified monitoring and optimization
- New invention can “bend” MRC curves
- Further performance improvements
  - Significant improvement, even for single workload!
  - Bigger wins for mixed and partitioned workloads
- Ongoing, in-progress R&D

Next-Gen Curve-Bending AI

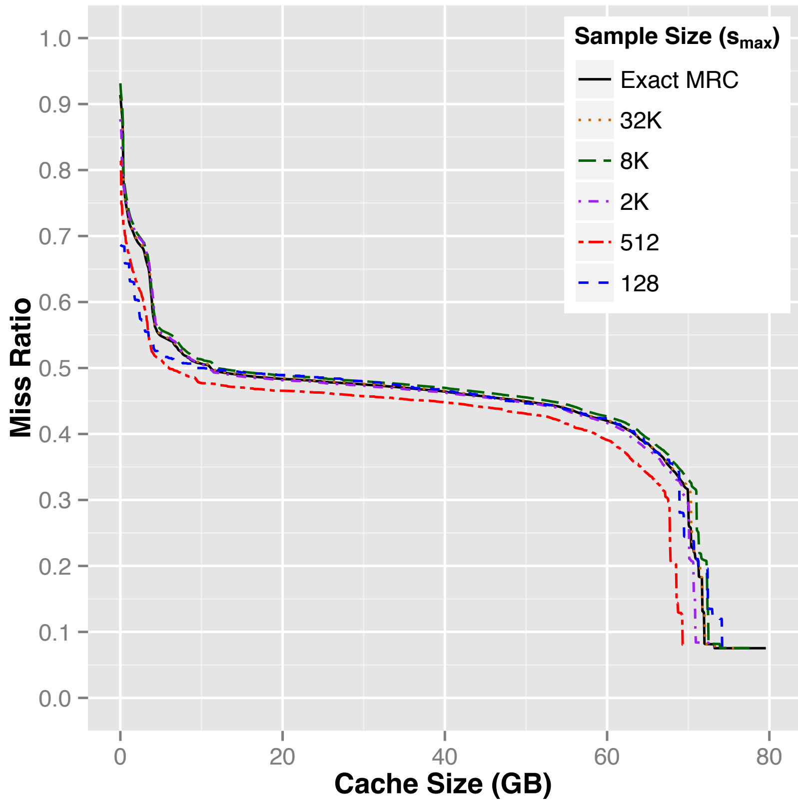


Conclusion

- Miss Ratio Curves (MRCs): game-changing storage tool
- CloudPhysics' MRC algorithm = up to 10,000x improvement
- Online MRCs now practical:
  - ~20 million IO/s per core; amortized 60 ns per IO
  - High accuracy in 1 MB
  - Feasible for for memory-constrained firmware, drivers
- Looking to partner with storage systems vendors
- Applications:
  - Workload-aware predictive cache sizing & tuning
  - Software-driven cache partitioning for "free" performance
  - Latency / Throughput guarantees via cache QoS

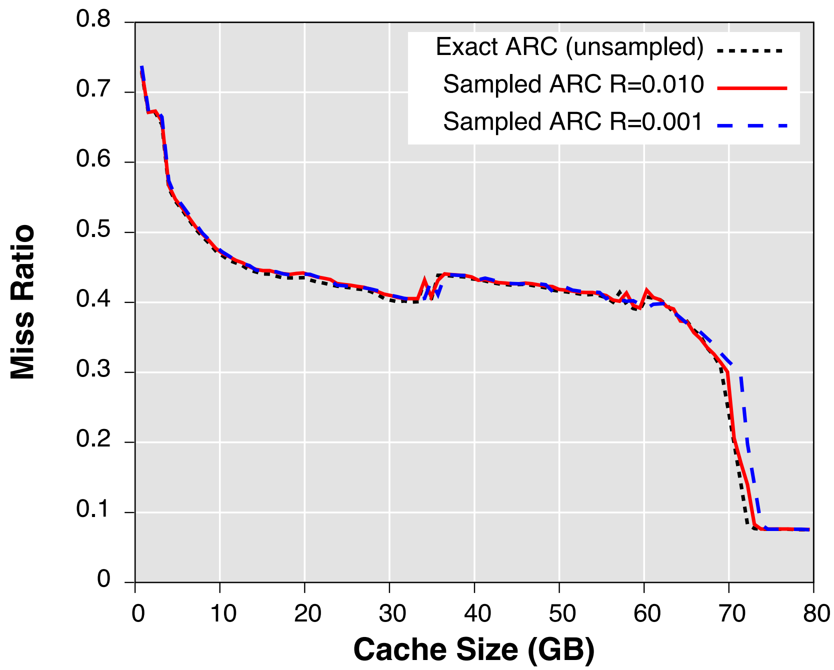
Appendix

Example SHARDS MRCs



- Block I/O trace *t04*
  - Production VM disk
  - 69.5M refs, 5.2M unique
- Sample size s_{max}
  - Vary from 128 to 32K
  - $s_{max} \geq$ 2K very accurate
- Small constant footprint
- SHARDS_{adj} adjustment

Generalizing to Non-LRU Policies



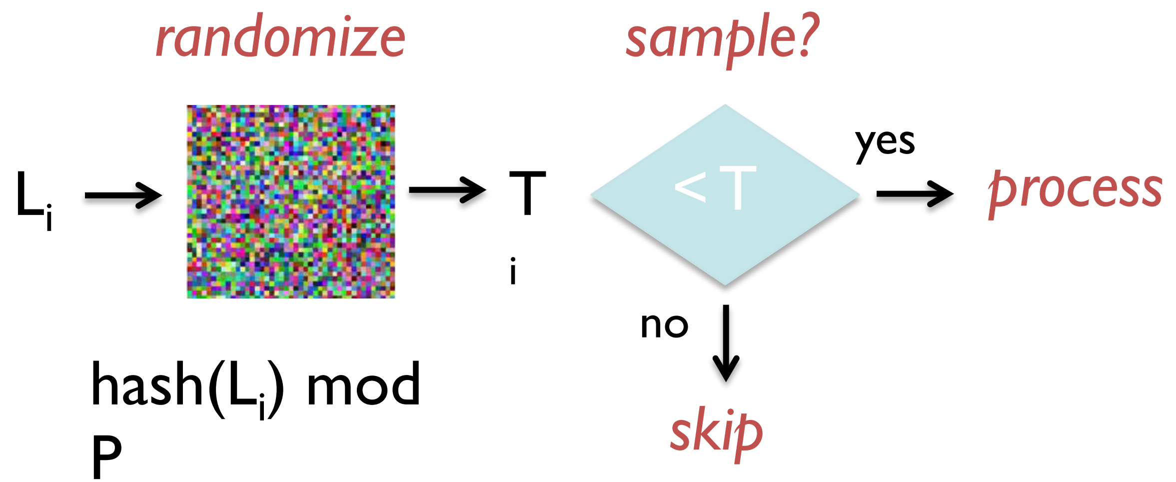
- Sophisticated algorithms
  - ARC, LIRS, Clock-Pro, …
  - No single-pass methods!
- Scaled-down simulation
  - Hashed spatial sampling
  - Simulate each size separately
- Example ARC results
  - 100 different cache sizes
  - 0.01 MAE with R = 0.001
  - 1000× memory reduction

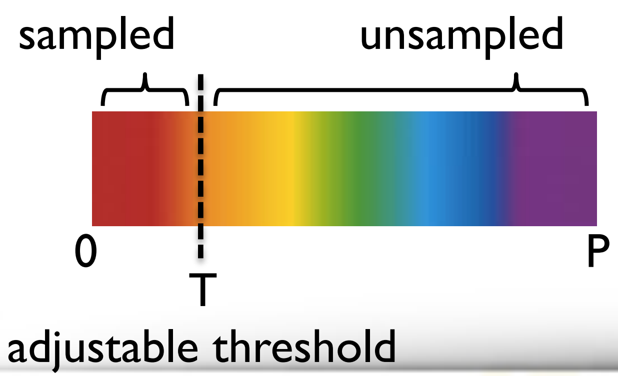
Mattson Algorithm Example

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

- Reuse distance
  - Unique refs since last access
  - Distance from top of LRU-ordered stack
- Hit if distance < cache size, else miss

Spatially Hashed Sampling

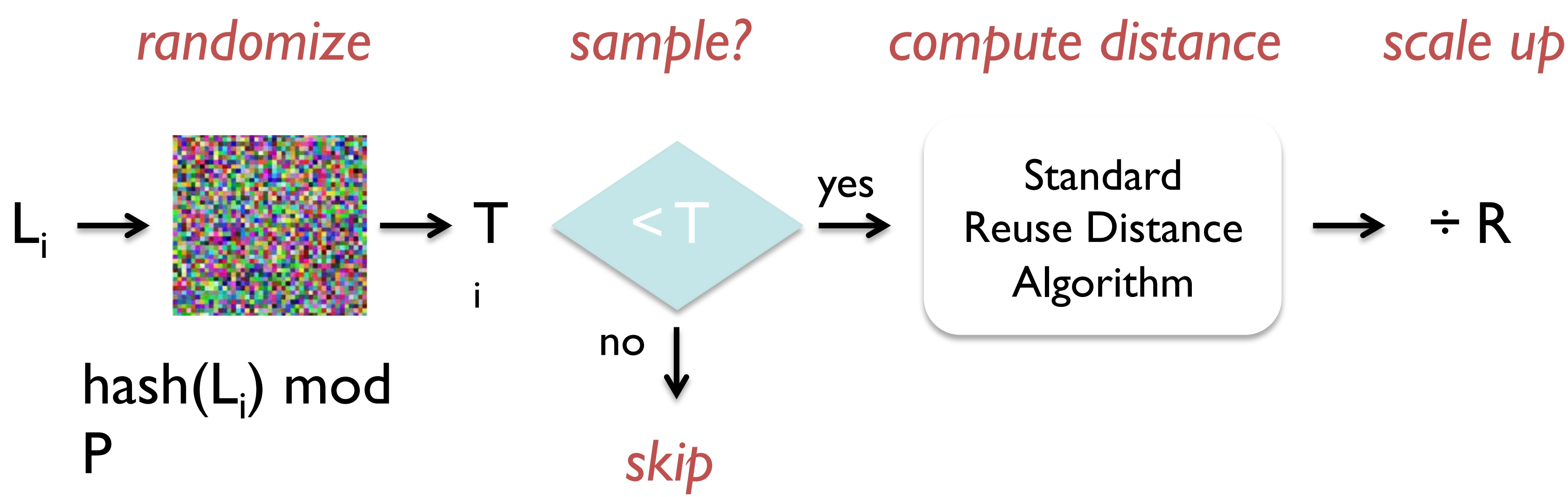




sampling rate R = T / P

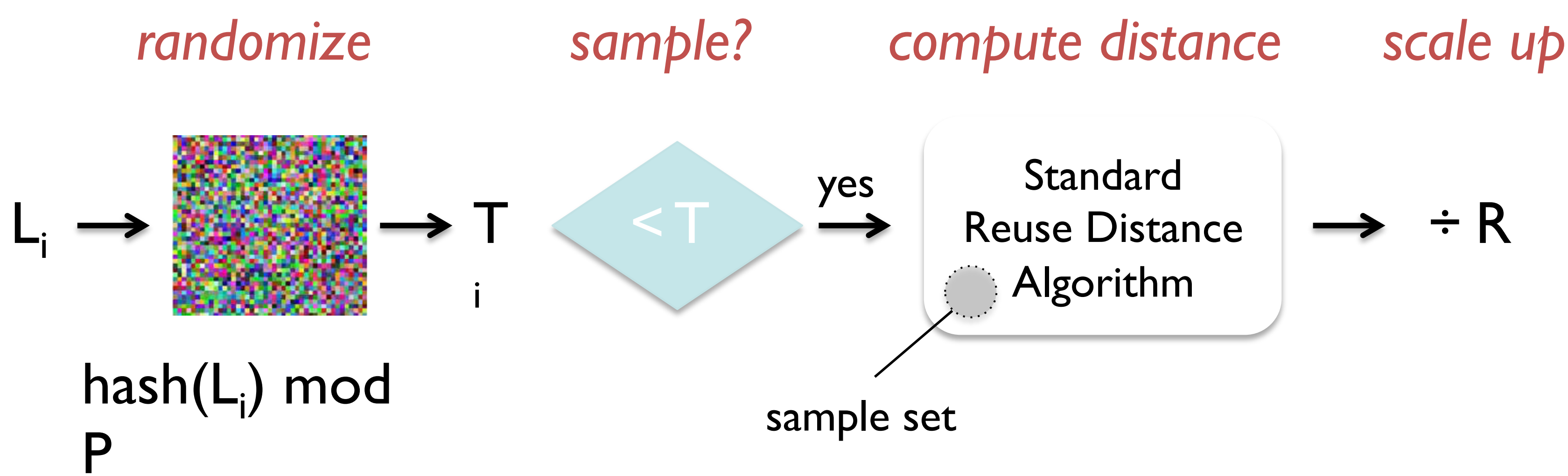
subset inclusion property maintained as R is lowered

Basic SHARDS

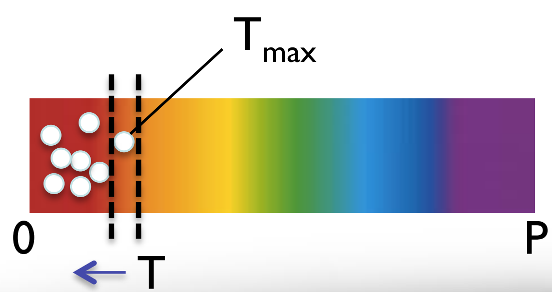


Each sample statistically represents 1/R blocks
Scale up reuse distances by same factor

SHARDS in Constant Space

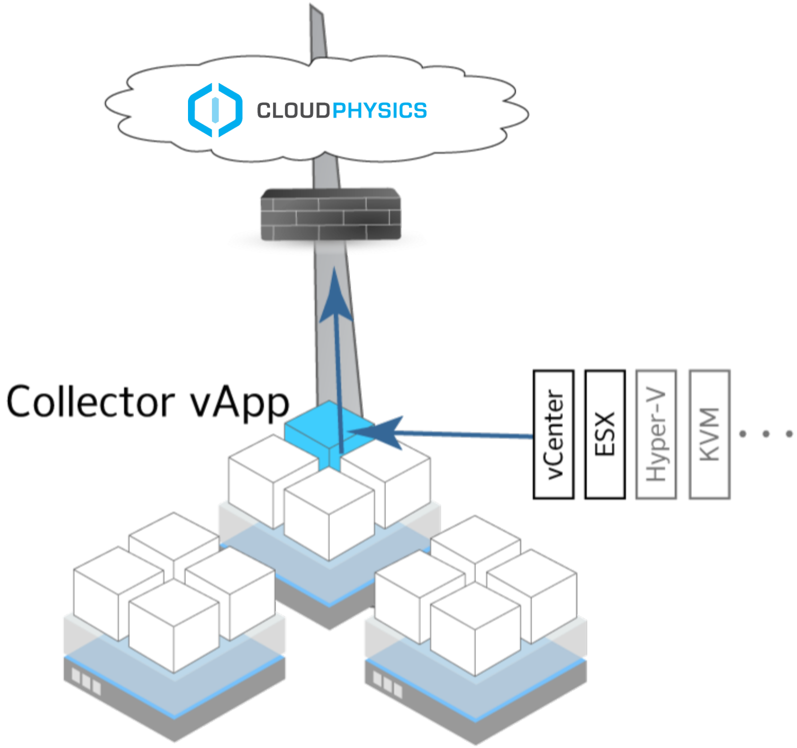


evict samples to bound set size



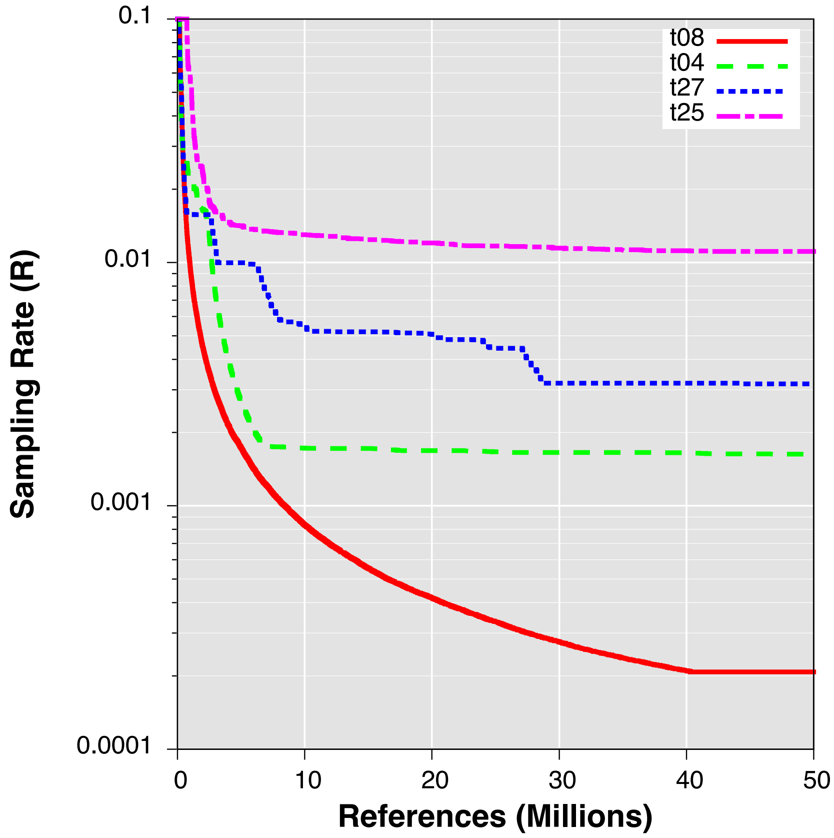
lower threshold $T = T_{max}$
reduces rate $R = T / P$

Experimental Evaluation



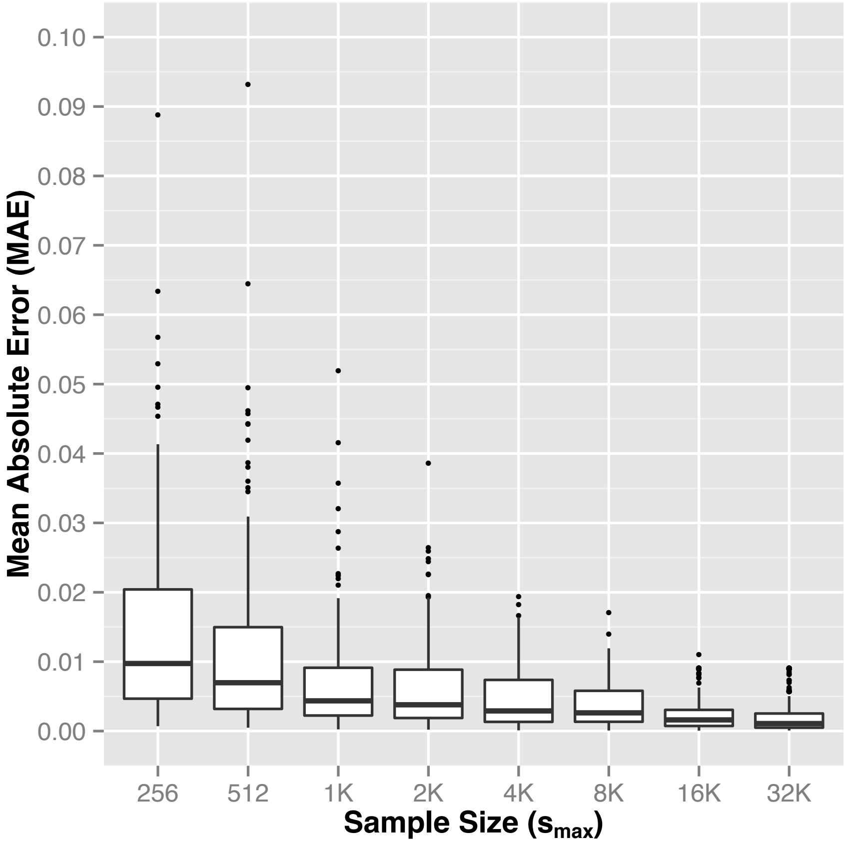
- Data collection
  - SaaS caching analytics
  - Remotely stream VMware vscsiStats
- 124 trace files
  - 106 week-long traces CloudPhysics customers
  - 12 MSR and 6 FIU traces SNIA IOTTA
- LRU, 16 KB block size

Dynamic Rate Adaptation



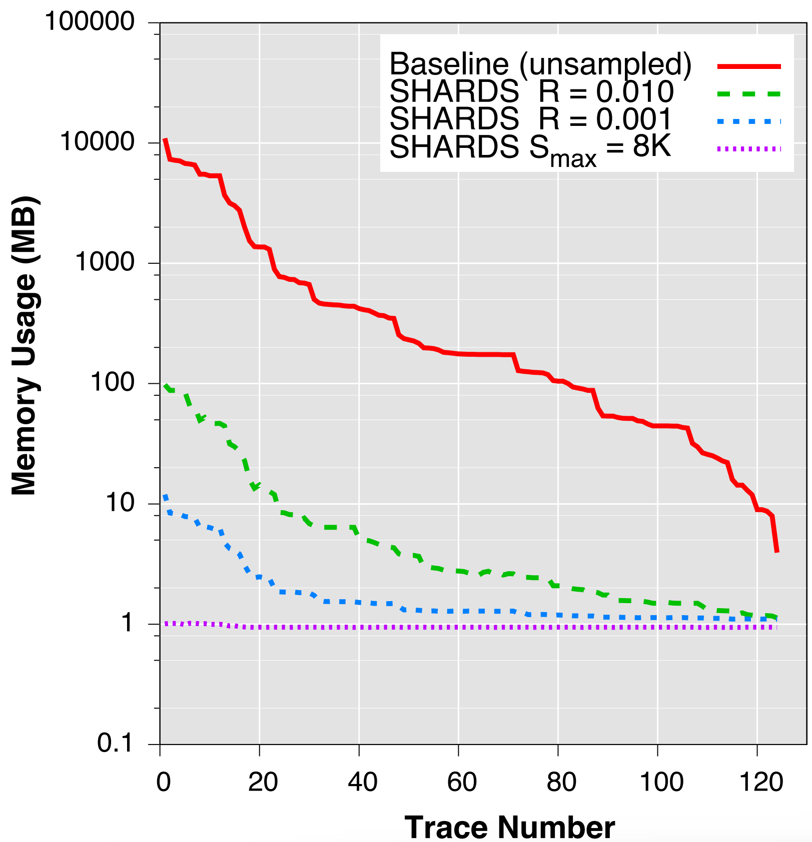
- Adjust sampling rate
  - Start with $R = 0.1$
  - Lower R as M increases
  - Shape depends on trace
- Rescale histogram counts
  - Discount evicted samples
  - Correct relative weighting
  - Scale by R_{new} / R_{old}

Error Analysis



- Mean Absolute Error (MAE)
  - |exact – approx|
  - Average over all cache sizes
- Full set of 124 traces
- Error $\propto 1 / \sqrt{s_{max}}$
- MAE for s_{max} = 8K
  - 0.0027 median
  - 0.0171 worst-case

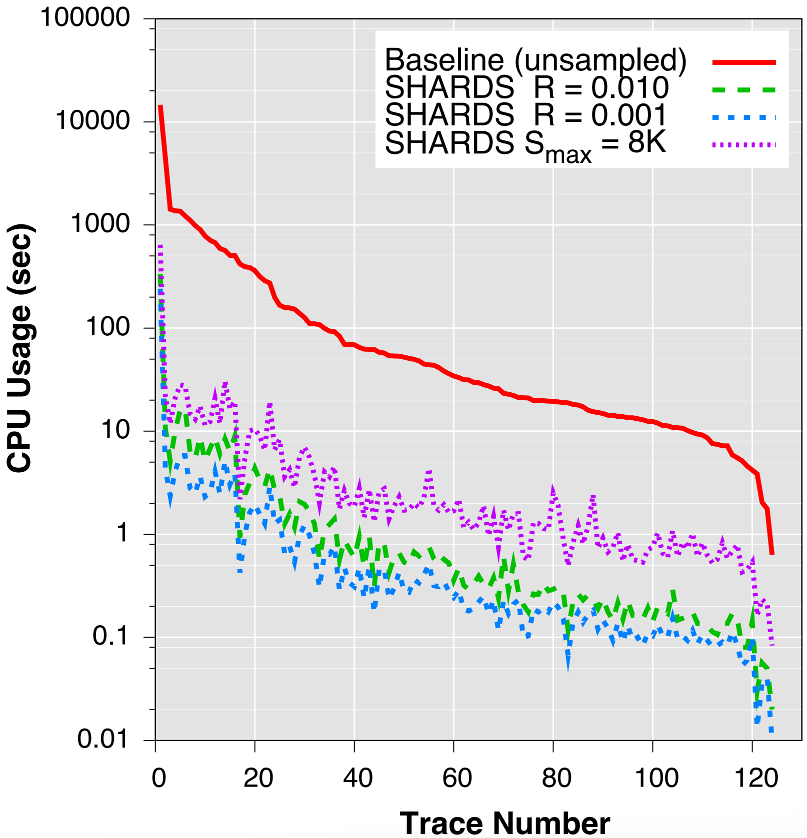
Memory Footprint



- Full set of 124 traces
- Sequential PARDA
- Basic SHARDS
  - Modified PARDA
  - Memory ≈ R × baseline for larger traces
- Fixed-size SHARDS
  - New space-efficient code
  - Constant 1 MB footprint

SDC 15

Processing Time



- Full set of 124 traces
- Sequential PARDA
- Basic SHARDS
  - Modified PARDA
  - R=0.001 speedup 41–1029×
- Fixed-size SHARDS
  - New space-efficient code
  - Overhead for evictions
  - S_{max}= 8K speedup 6–204×

SDC 15