
# Your Cache is Overdue a Revolution: MRCs for Cache Performance and Isolation

Irfan Ahmad, CTO, CloudPhysics

# 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.

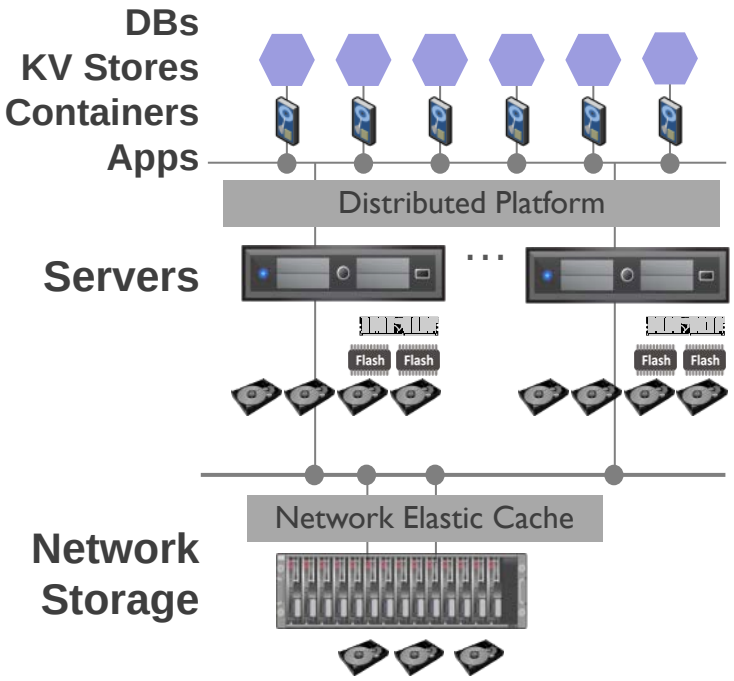
NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

# Abstract

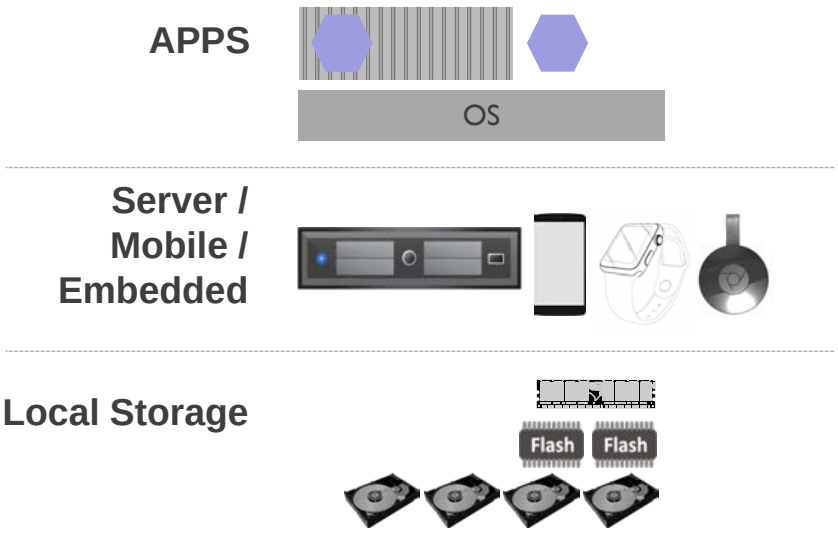
## Your Cache is Overdue a Revolution: MRCs for Cache Performance and Isolation


- Recently, a new, revolutionary set of techniques have been discovered for online cache optimization. Based on work published at top academic venues (FAST '15 and OSDI '14), we will discuss how to 1) perform online selection of cache parameters including cache block size and read-ahead strategies to tune the cache to actual customer workloads, 2) dynamic cache partitioning to improve cache hit ratios without adding hardware and finally, 3) cache sizing and troubleshooting field performance problems in a data-driven manner. With average performance improvements of 40% across large number of real, multi-tenant workloads, the new analytical techniques are worth learning more about.

# Cache as a Layer of Control



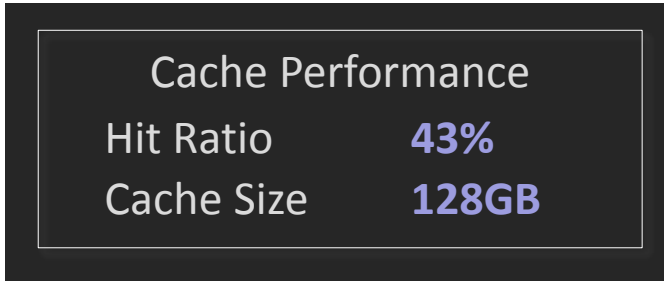

Cloud Scale Data Caching




Single System Data Caching

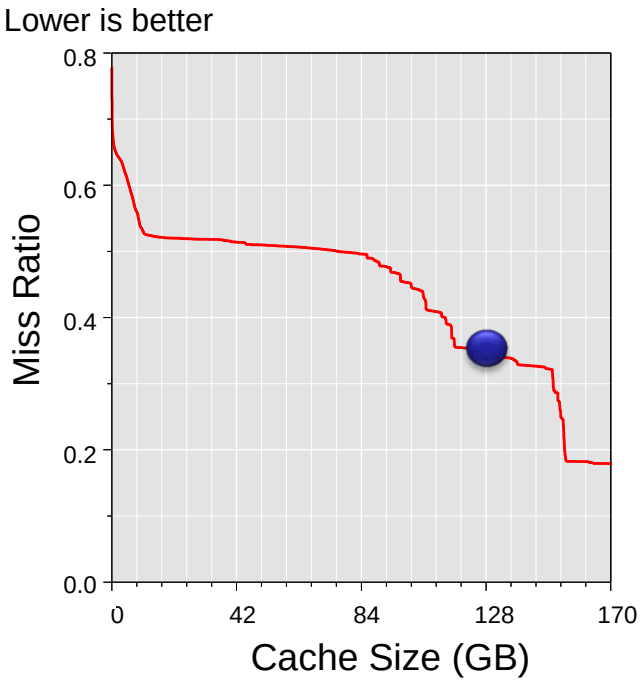
**Goal: Customer Self-Serve on Sizing, Tuning, Troubleshooting**
**Goal: 50%+ Hardware Efficiency Gain**
**Goal: SLO guarantees, Latency & Throughput**

# Cache Performance Questions Unanswered



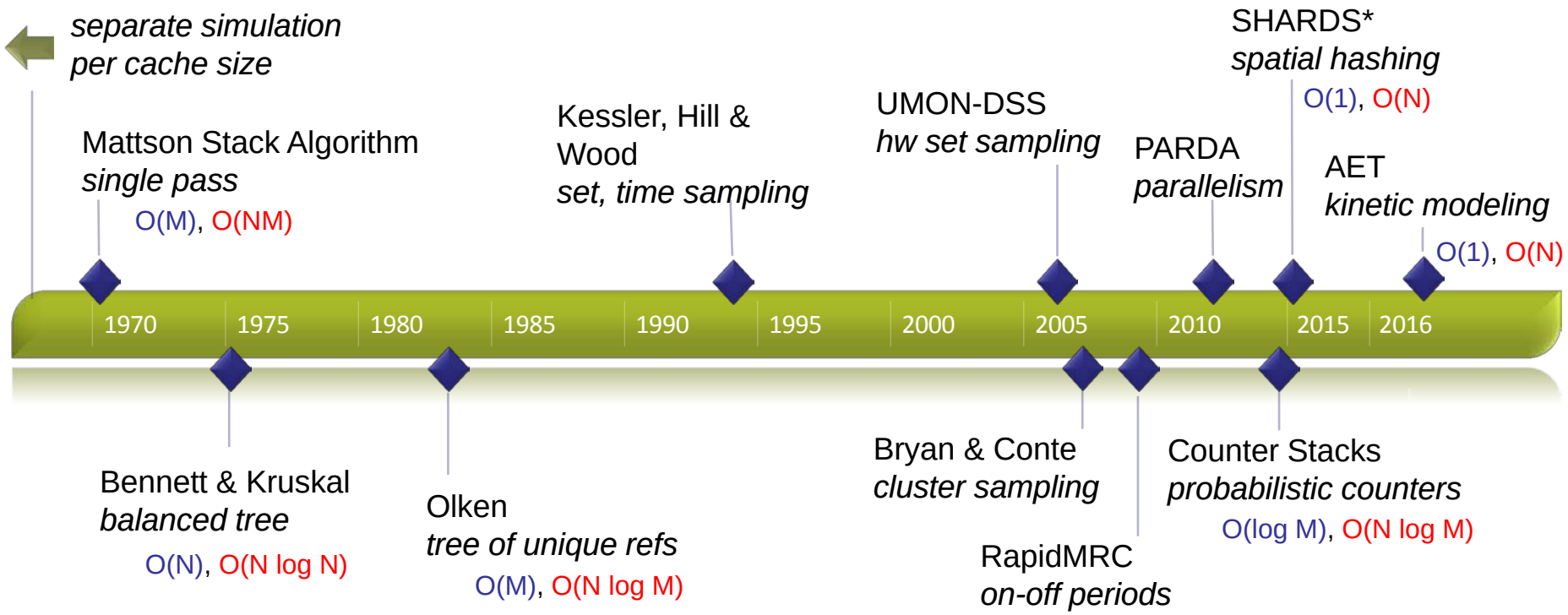

- Is this performance good? Can it be improved?
- What happens if I add / remove some cache?
- What if I add / remove workloads?
- Is there cache thrashing / pollution?
- What if I change cache algorithm parameters?

# 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

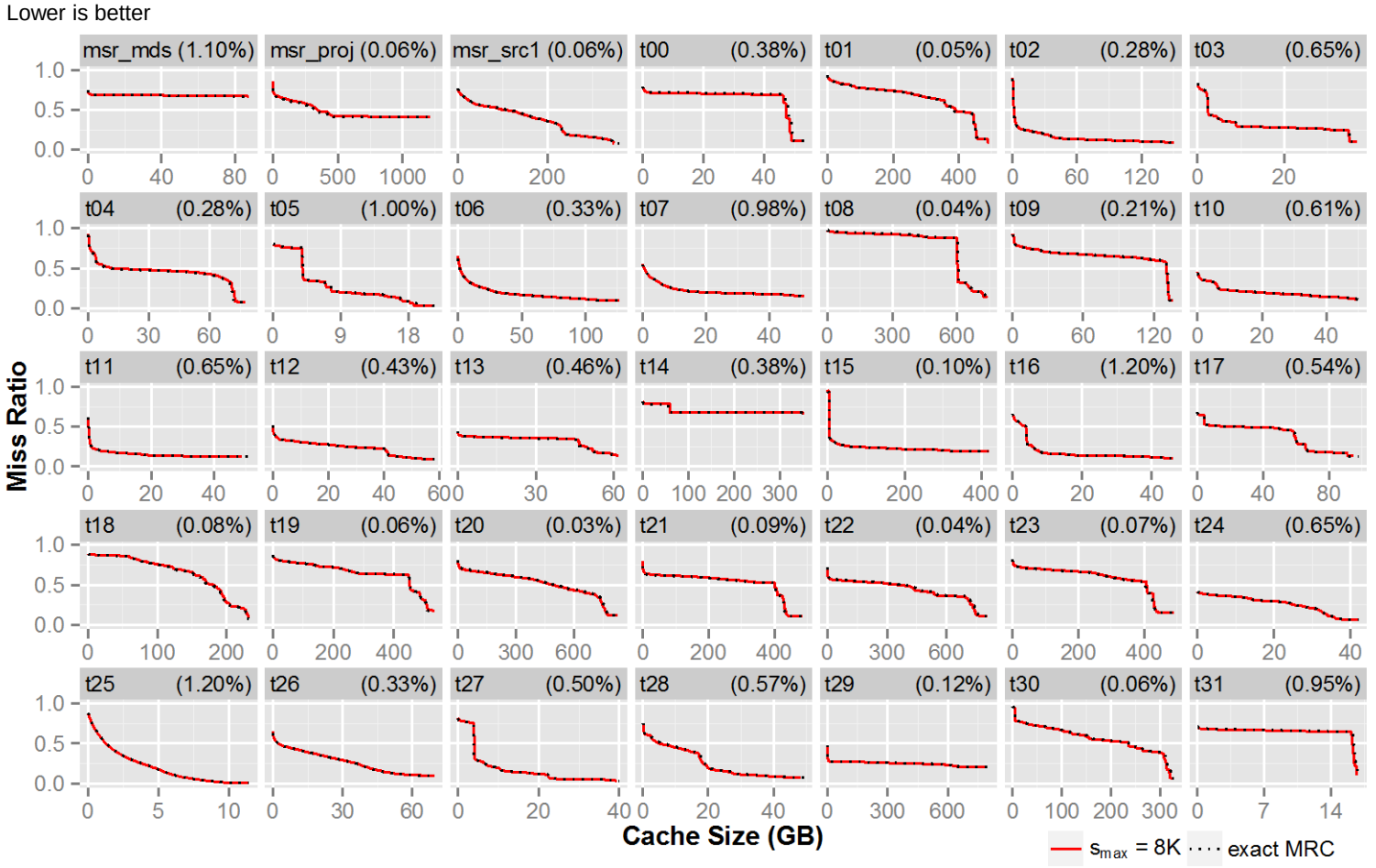
# MRC Algorithm Research




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

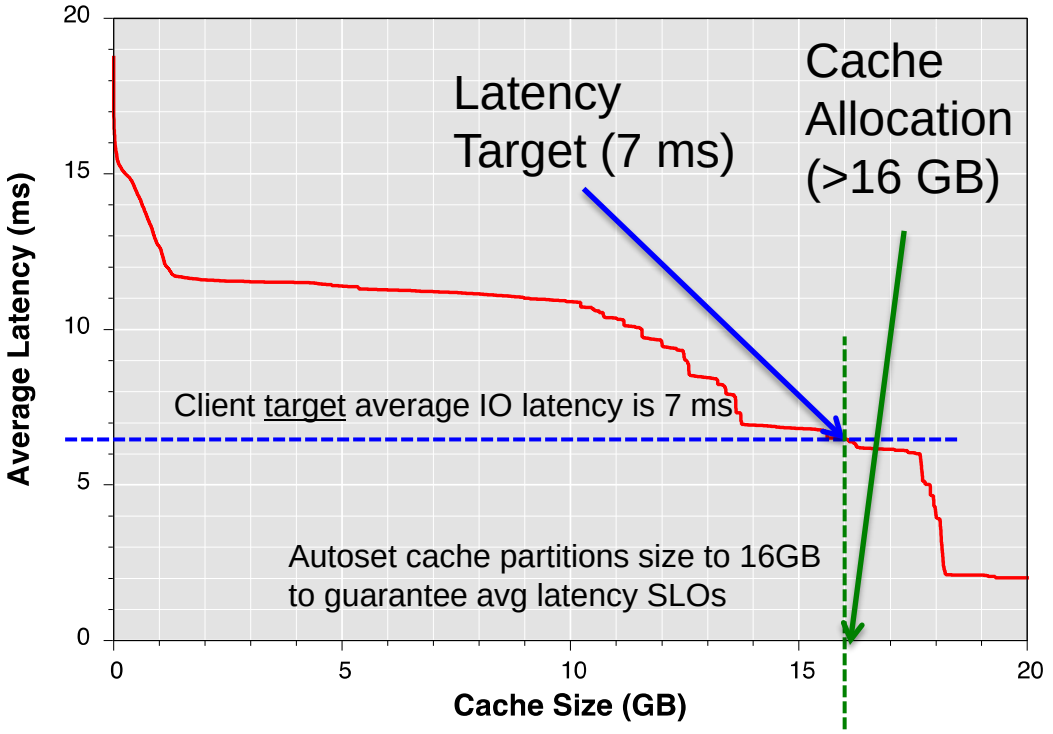
* Spatially *H*ashed *A*pproximate *R*euse *D*istance *S*ampling

# MRCs from Production Workloads



# Achieving Latency Targets

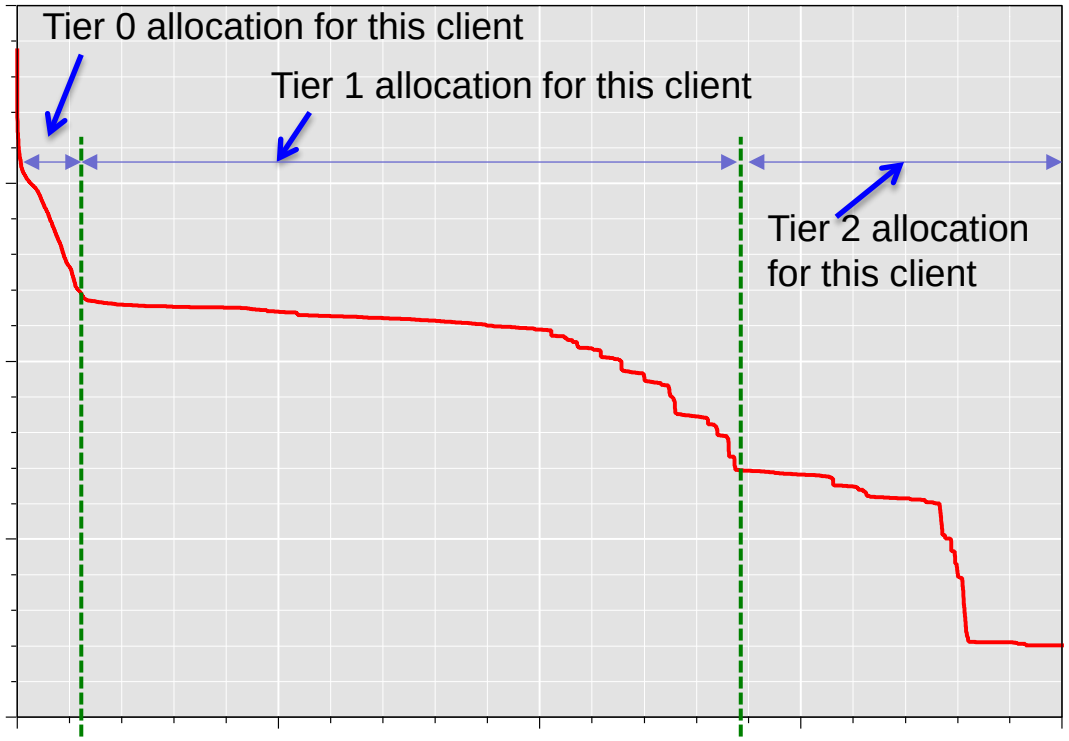
- Convert to a Latency-Cache Curve
- Set client cache partition sizes to provide *average latency* SLOs
- Can set targets for %ile latency, not just avg (e.g. 75th %ile, 95th %ile)
- Drive per-client *tier* occupancy sizing for latency SLOs




* Throughput targets can be implemented similarly

# Per-Client Multi-Tier Sizing

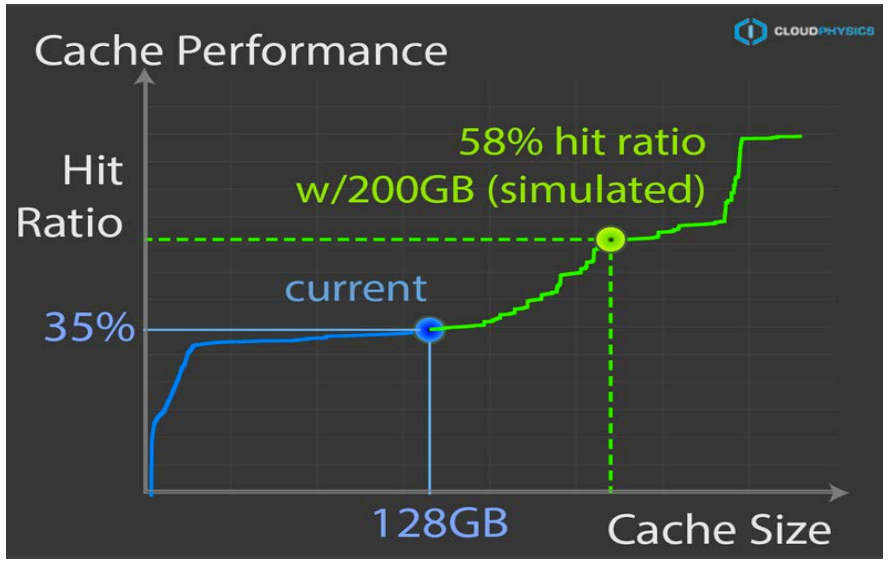
- MRCs give strong guidance on sizing per-client tier allocations
- Latency guarantees can be achieved by computing hit/miss costs (latency) of the tiers against hit/miss ratios
- Each tier can be multi-tenant using sizing via MRCs
- Model network bandwidth as a function of cache misses from each tier



# Self-Service Monitor / Size / Troubleshoot

- Tune and optimize production workloads
- Show MRCs in monitoring UI
  - Ops teams can troubleshoot or self-service on sizing
  - Size caches in production, trigger alerts, etc.

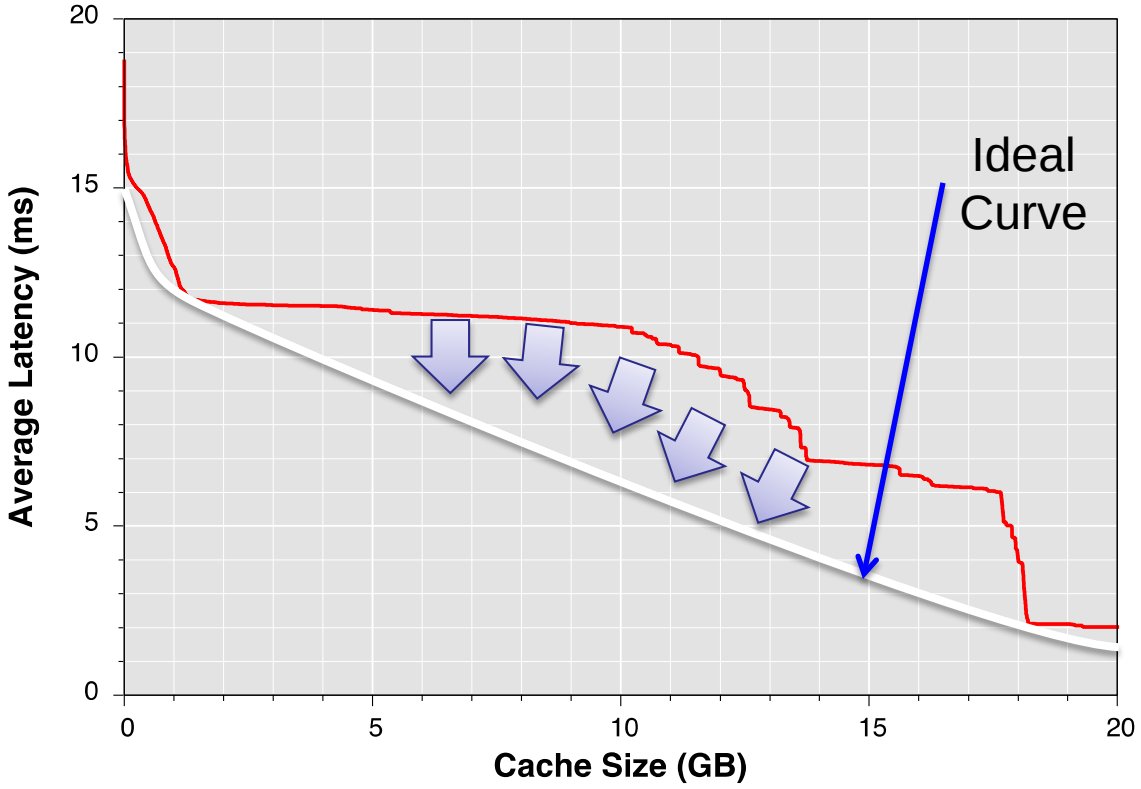
Higher is better




Interactive what-if analysis of effect to a workload of more or less cache

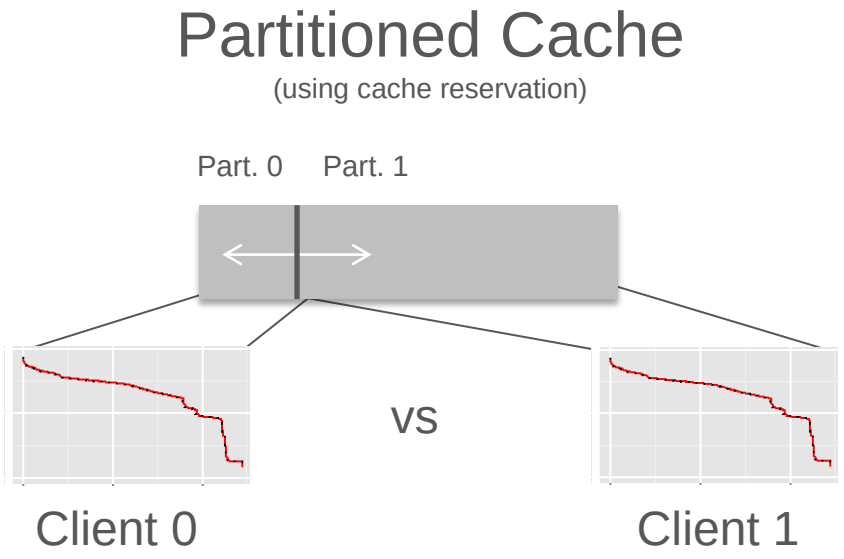
# Performance Gain Thrash Remediation

- New mechanism allows online, optimal MRC bending via a thrash remediation algorithm
- Optimize cache for cache-unfriendly workloads
- Lower miss rates even for single workloads
- Compatible with partitioning for additional benefit



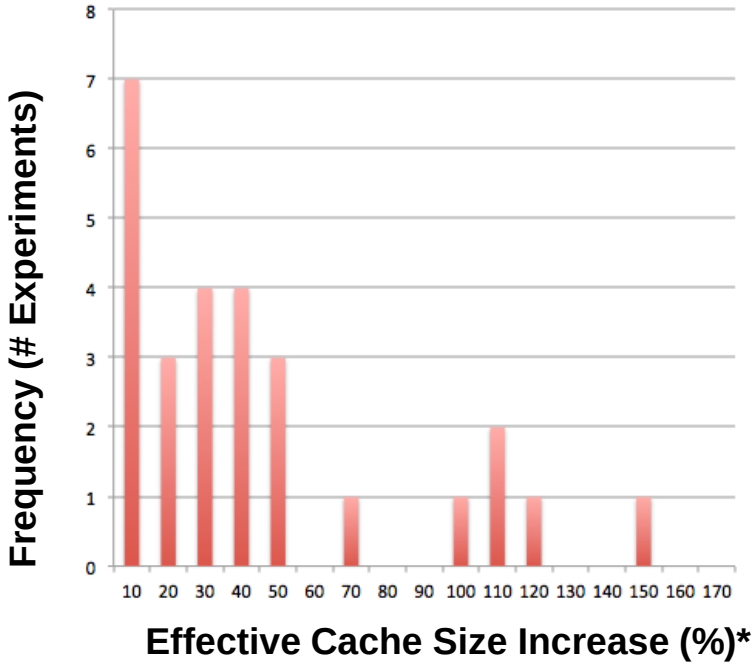
# Performance Gain From Partitioning

- Improve aggregate cache performance
  - Allocate space based on *client benefit*
  - Prevent inefficient space utilization / thrashing
- Mechanism: Partition cache across clients
  - Compute per-client MRCs cheaply
  - Isolate and control competing clients, LUNs, VMs, tenants, DB tables, etc.
  - Optimize partition sizes using MRCs
- Adapt to changing workload behavior
- Certain SDS platforms already support Partitioning



# Performance Gain From Partitioning (Results)

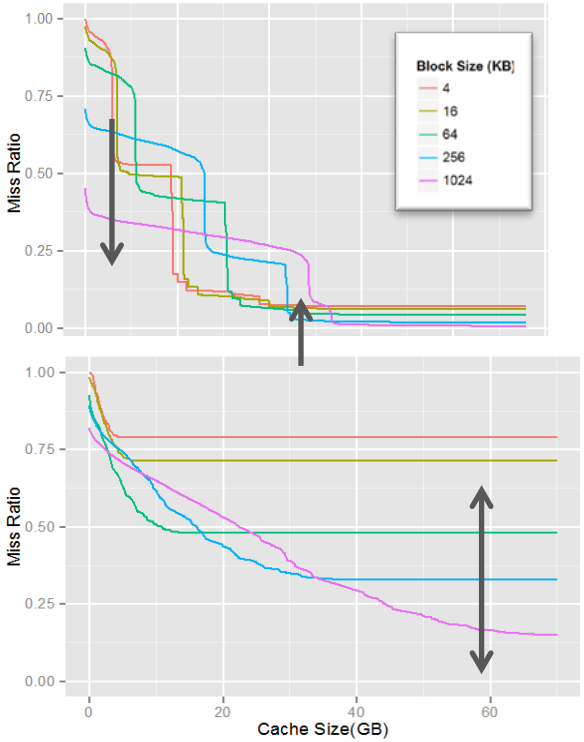
- MRC-guided partitions vs. global LRU
- Effective cache size
  - 40% larger (avg)
  - 146% larger (max)
  - Computed as the %age larger cache hardware needed whose performance would match MRC-guided partitioning.
  - Metric reflects the capital expenditure avoided to get higher performance by using a software-only technique.



# Performance gain from Auto Cache Policy Tuning

- Quantify impact of parameter changes
  - Cache block size, use of sub-blocks, Write-thru vs. write-back, Size of Read vs Write cache, replacement policy…
- Explore without modifying actual production
  - Simulate multiple configurations concurrently
  - Multiple MRCs, each with different parameters
- Dynamic online optimization
  - Determine best configuration
  - Adjust actual cache parameters
- Tuning parameters impact performance dramatically (can be 100%+)
- Parameter selection can be automated

Two examples show workload-sensitive cache-block-sensitive performance curves



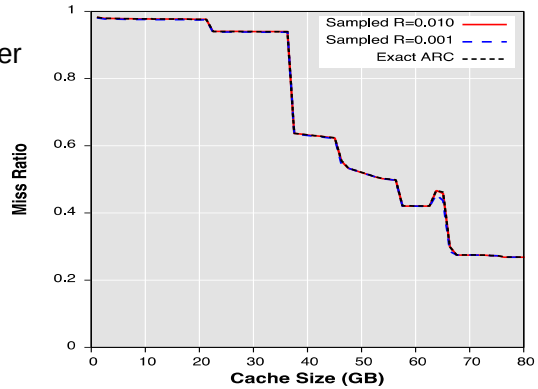

Can automatically pick the correct parameter settings via online MRCs.

# Generalizing to Non-LRU Policies (Special to SHARDS [FAST 2015])

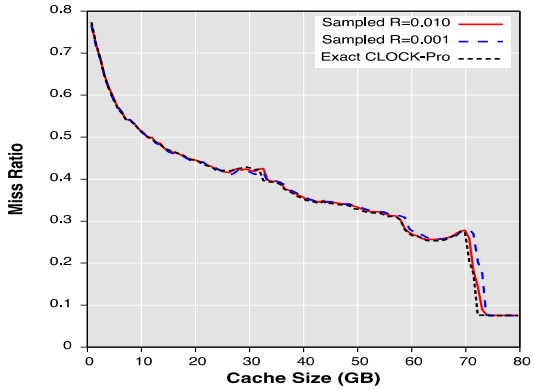
- Sophisticated caching algorithms
  - ARC, LIRS, CAR, Clock-Pro, 2Q, …
  - No known single-pass methods!
- Scaled-down simulation
  - Leverages SHARDS hashed spatial sampling algorithm
  - Simulate each size separately
- Still highly efficient
  - Low sampling rate R = 0.001
  - 1000× reduction in memory,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

Lower is better

ARC
MSR-Web Trace



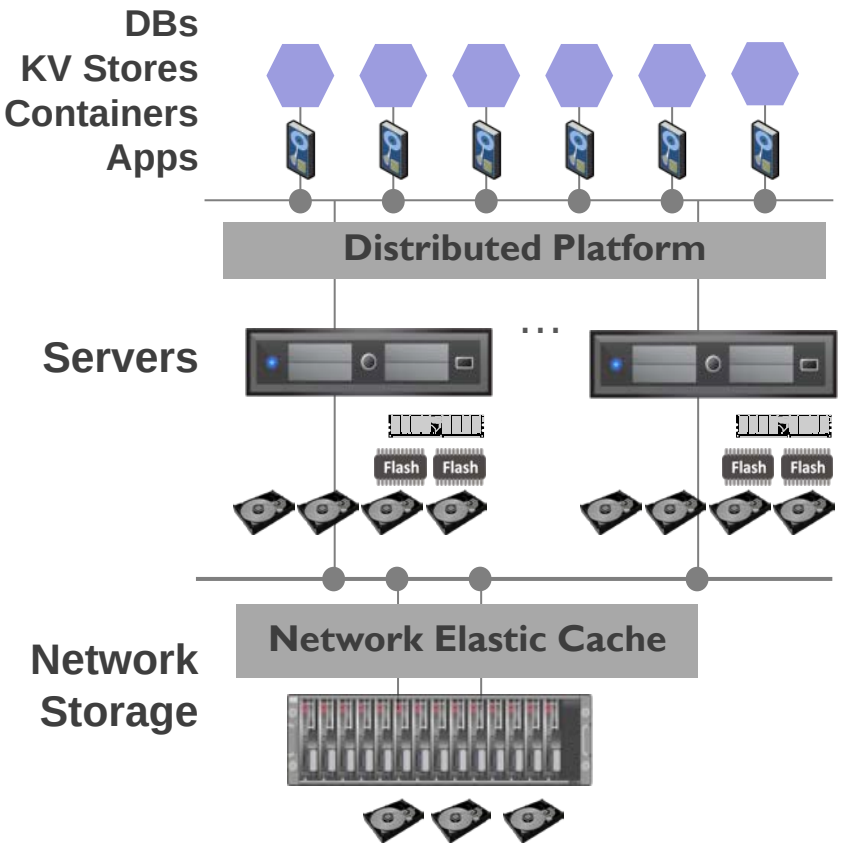

CLOCK-Pro
Trace t04



# Systems Implementation

- Easy integration with existing embedded systems
- Example C interface
  - `void mrc_process_ref(MRC *mrc, KEY opaque);`
  - `void mrc_get_histo(MRC *mrc, Histo *histo);`
- Extremely low resource usage (SHARDS example)
  - Accurate MRCs in <1 MB footprint
  - Single-threaded throughput of ~20M blocks/sec
  - Average time of `mrc_process_ref()` call ~50 ns
  - No floating-point, no dynamic memory allocation

# Cache Resource Management in Software Defined Storage




## New Features possible with Miss Ratio Curves (MRCs)

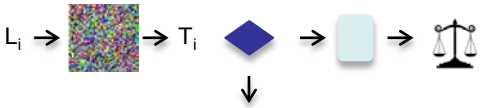
- Distributed Multi-tenant Policies
- Global Latency Guarantees
- Global Throughput Guarantees
- Flash Cache Auto-Reservation
- DRAM Cache Allocation
- Cache Time Series Analysis
- Cache Thrashing Remediation
- Remote v Local Cache Allocation
- Write Cache Allocation
- Self-tuning Cache Policies
- Automatic Write Policy Selection
- Automated Tiering

# Multi-Tenant Performance Challenges

- Allocate resources to get biggest bang for the buck
  - Equalize marginal value of additional cache block to each workload
  - Support weighted value based on SLO goals, importance, price
  - Similar to existing resource management techniques for CPU and memory
  - Cluster-wide cache resource management extremely valuable
- Today, global LRU or global ARC commonly used
  - Can exhibit gross unfairness, interference and unpredictability
  - Vulnerable to thrashing/scan pollution with multi-tenancy
- How to determine marginal returns of additional cache?
  - MRCs solve this problem by providing the entire utility curve
  - But prior methods too expensive to construct online (both time and space)
- Lightweight MRCs enable dramatic cache-management benefits

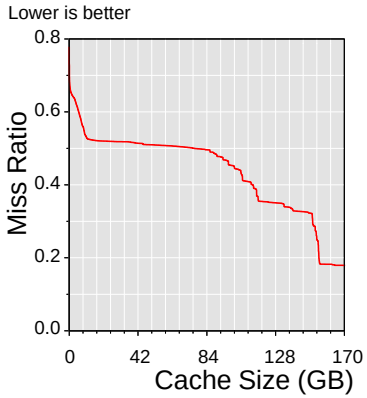
# Step-by-Step Recap

## 1. Calculate Miss Ratio Curve (MRC)



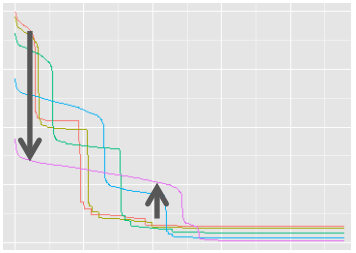

Trivial integration with cache code base. In case of SHARDS, single new function call in I/O path (~ 50 ns).

## 2. Display Cache Utility




Trivial integration with monitoring code base.

## 3. Auto-Tune Cache Parameters

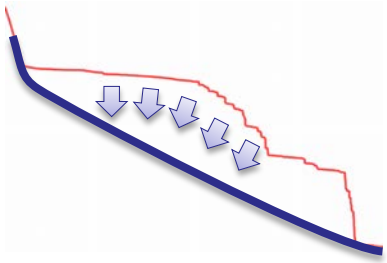


Small modifications to cache code base. Auto-select cache parameters using multiple online MRCs.

Performance improvements (often see 50%+ opportunity)

# Step-by-Step Recap

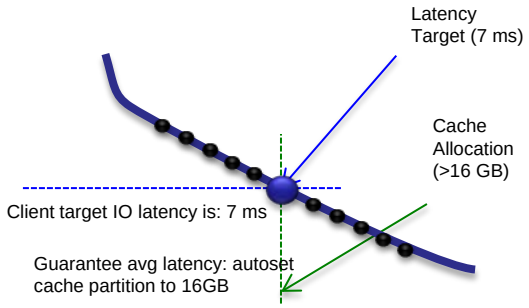
## 4. Thrashing Remediation



Thrashing: a workload fills cache with entries not reused. Small modifications to cache code base to selectively prefer some blocks during eviction.

Not uncommon to see 50% performance improvement from this step.

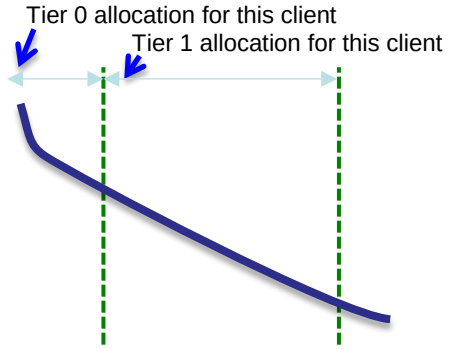
## 5. Latency Guarantees




Trivial arithmetic to convert miss ratio curve to a latencies-cache curve. Then a simple control loop to set cache size to match SLO.

First predictive latency/throughput guarantee system.

## 6. Accurate Tiering Decisions

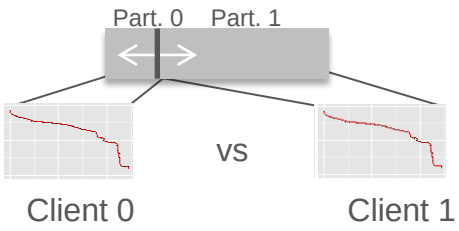



A new cache allocation policy manager in caching code base.

Cluster-wide allocation of cache resources to match SLOs. For example, to achieve some 500 µs, allocate 5GB local DRAM, 5GB remote DRAM, 15 GB local SSD, 20 GB remote SSD.

# Step-by-Step Recap

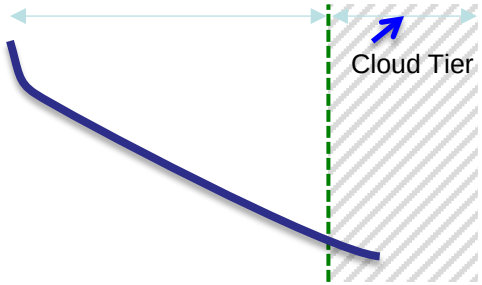
## 7. Multi-Tenant Partitioning




Small modifications to cache code base to create virtual partitions.

Our data shows an average cache efficiency improvement of ~50%.

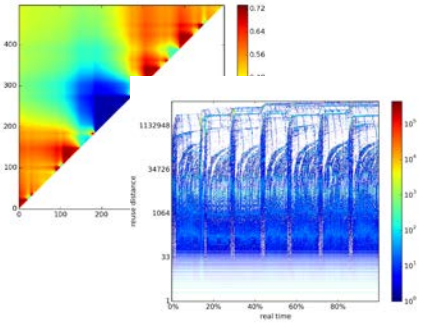
## 8. Cloud Storage




Use the latencies-cache curve to size the cloud tier.

Enable a data-driven, predictive sizing of the cold data set in a remote tier (for ROBO, edge, colo use cases)

## 6. Cache Time-Series MRCs



Analytics to predict caching cost-benefit analysis over time.

Enables automated and predictive cache sizing using time series analysis.

# Attribution & Feedback

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

## Authorship History

**Irfan Ahmad / 2016-09-22**

## Additional Contributors

**Carl Waldspurger**

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