SANCache

Performance Boosting & Workload Isolation in Storage Area Networks

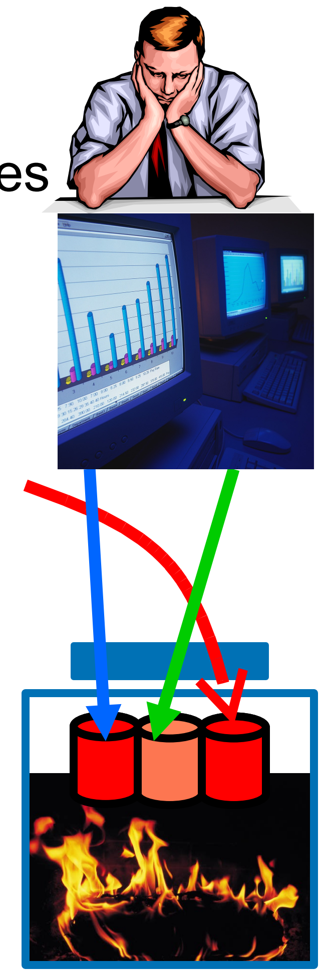
Ismail Ari, Melanie Gottwals, Dick Henze

HP Labs, Palo Alto, CA

IEEE Mass Storage Systems and Technologies 2006

Problems in Current Data Centers

- Problem
  - Fast, dynamic, cost-competitive business environment
  - Continuous (24x7) pressure on computing infrastructures
  - Flash (surges of) demand for high system performance
  - CPUs are fast, **disk-based storage is the bottleneck**
- Current business practice to reduce costs
  - Consolidate/Pool resources physically
  - Allow resource sharing → increase utilization
  - Simplify management
- End Result
  - Workload consolidation of different customers
  - Unpredictable performance



SANCache Problem Statement

I/O-intensive workloads can overwhelm disk-based storage

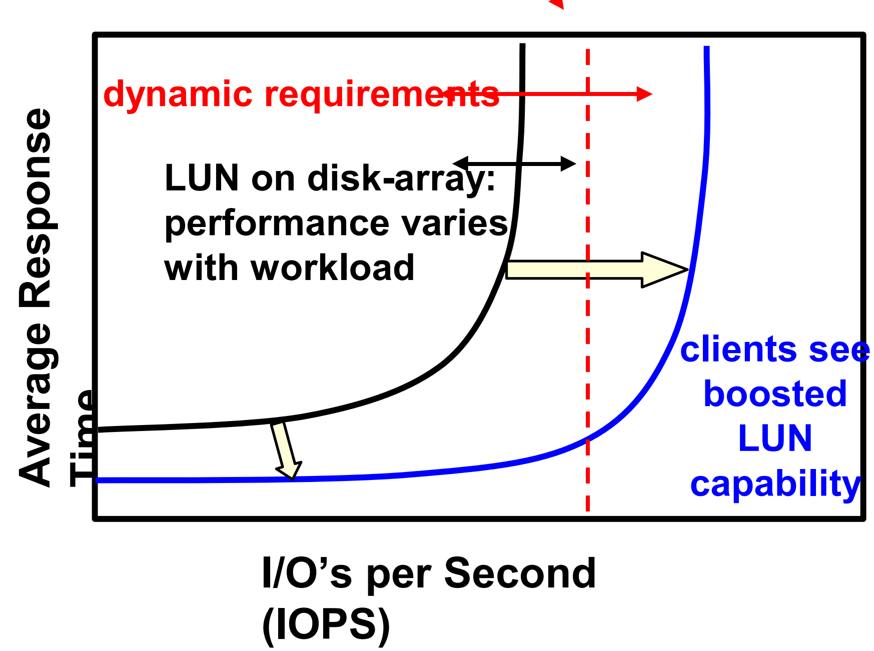
- Unpredictable application performance due to workload variations & interference
  - Difficult to isolate workload streams or guarantee performance
  - Example: database storage serving concurrent OLTP, Decision Support, and Backup phases in a 24 x 7 enterprise
- Throughput & access-density **[IOPS/GB]** limitations of disks
  - Disk capacity is over-provisioned to meet IOPS goals, which increases space, power, and management cost
- Solution? Static partitions in solid-state-disk (SSD) storage?
  - **No!** Manual configurations are management intensive and they do not optimize utilization of this costly resource under dynamic conditions

SANCache Solution

- Automatically allocates a SAN-wide cache resource across workloads to meet quality of storage service requirements:
  - "My OLTP transaction load requires a guaranteed 20,000 IOPS from 2 specific volumes for the next 40 minutes"

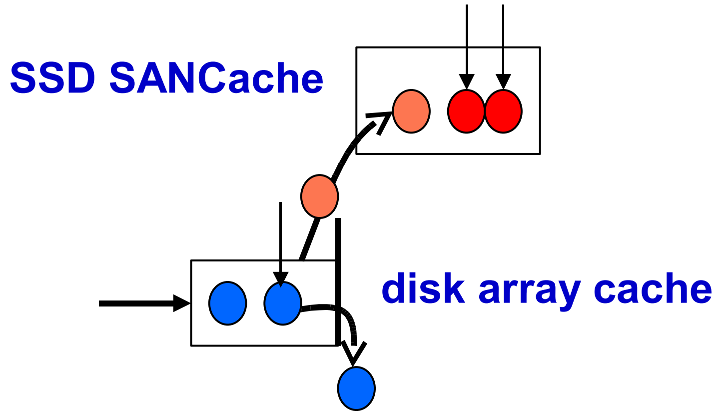
- A mechanism to dynamically augment disk-based storage in the SAN

- This feature is selectively enabled on a Logical Unit (LUN) basis



SANCache Technology Challenges

- Designing policies to select hot data
  - Across varied workloads and storage configurations
  - Adapting parameters to maintain heat of the cache contents
- Controlling migration rate
  - To avoid saturation of disk, array controller, or fabric resources
- Out-of-band implementations
- Allocating cache resource optimally
  - Across multiple LUNs to maximize utility
  - Tie-in to storage services



Policy goals:

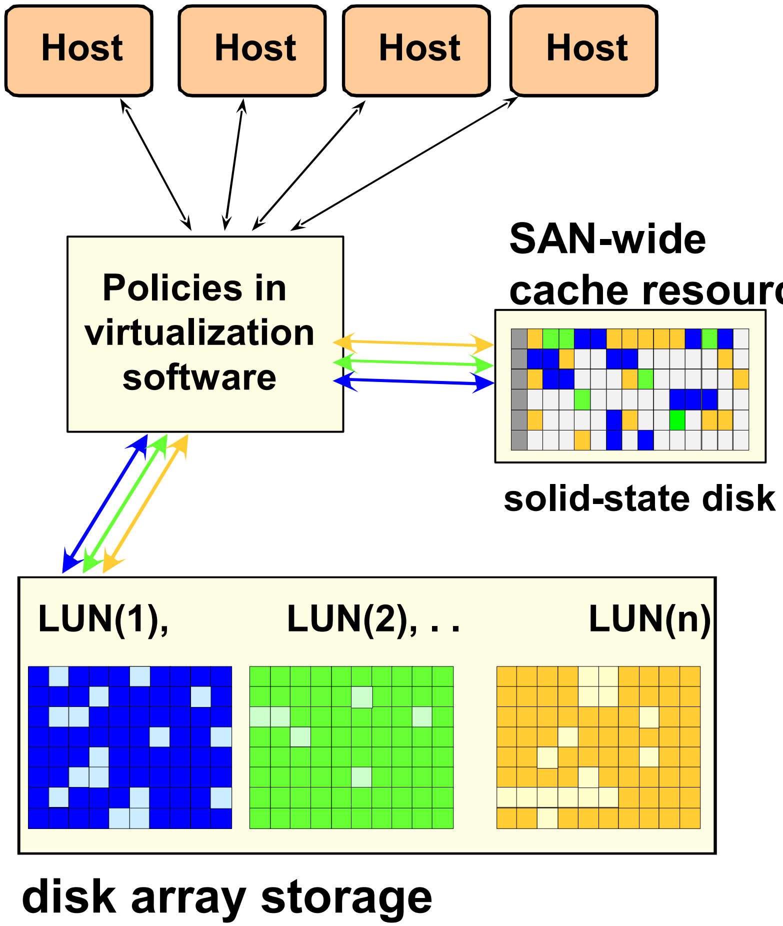
Achieve exclusivity

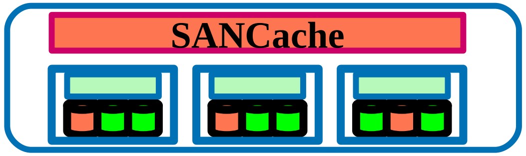
Maximize access density

Minimize migration overheads

SANCache Implementation

- For enabled LUNs, "hot" chunks of data are selectively migrated to a SAN-attached caching resource
- Existing host and disk-array resources remain unchanged
- Policies are implemented at the SAN fabric virtualization layer
- SANCache resource is provisioned independently from disk-array storage
- Current prototype uses an enterprise SSD connected to SAN





Design Requirements

- Justify Price($)/Performance (IOPS, MBps)
  - Maximize IOPS/GB = heat or access density
  - Has *temporal* (seconds) and *spatial* (GB) dimensions

- Automate hot data management
  - Too complex and time-consuming to be done manually
  - Requires continuous monitoring and tuning
  - ***Complexity*** = [#Applications] x [#OS] x [# H/W drivers]

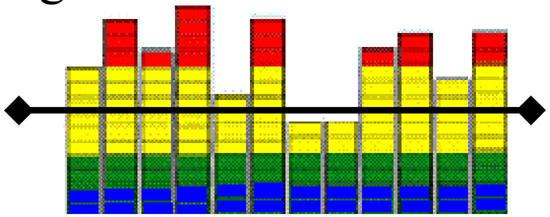
- Adapt: change caching policies and cache allocations
  - 1- Workload characteristics change and workloads mix
  - 2- Storage environment changes
  - 3- Performance requirements change on demand (SLA)

Design Requirements (Cont'd)

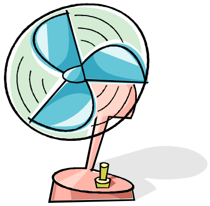
- Be useful in a heterogeneous (host-array) environment
  - Don't modify clients (hosts) and backend storage
  - Operate with different disk arrays (MSA, EVA, XP, FAB)
- Be online (24x7)
  - Addition/removal/modify policies when system is online
  - Do not adversely affect the foreground process
- Facilitate workload performance isolation
  - Enable boosting selectively for hosts, targets, LUNs

Design Features: Threshold-based Cache Placement

- Most Frequently Used (MFU) placement into SSD
  - Use static placement threshold → No magic value !?
    - So many storage configurations and dynamic workloads
  - Our adaptive threshold tries to match "best current static"
    - Performs better than any static when things change a lot



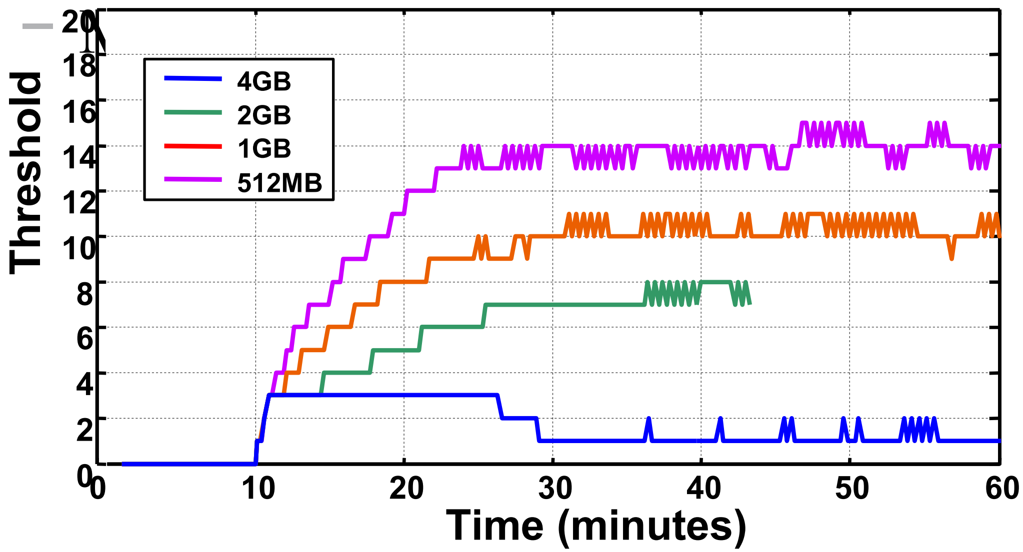
- Most *Recently* Frequently Used
  - Cool candidates for migration periodically
  - Selects currently hot chunks & moves less chunks



- Complements recency- & demand-based array cache
  - Short-term vs Longer-term policies

Adapt threshold

- Track changes and adapt *to maintain cache heat*
  - Changes in storage configuration and workloads
- Benefit-Cost Tradeoff **(BCRatio = Δhits/Δmigs)**
  - Benefit = **Δhits;** Cost = migrations (**Δmigs)**
- Example: adapting to different cache sizes



SPC-1 Workload Adaptive Threshold + Cooling

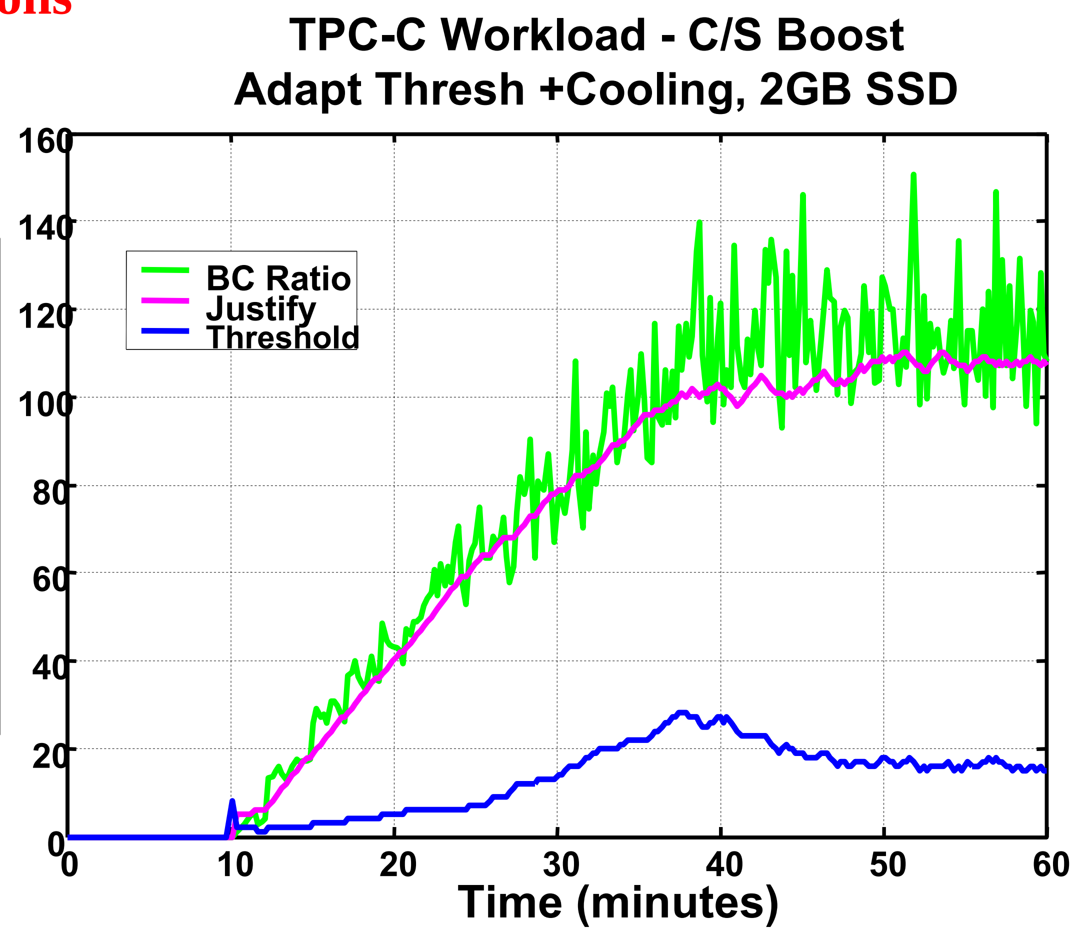
Adaptation Rule

Push to achieve higher BCRatio

Slow down migrations

```
if(ΔMigs > JUSTIFY) {
  if(BCRatio < JUSTIFY)
              THRESHOLD++
  elseif (BCRatio >= JUSTIFY)
              JUSTIFY++
} else {
              THRESHOLD--;
              JUSTIFY--;
}
```

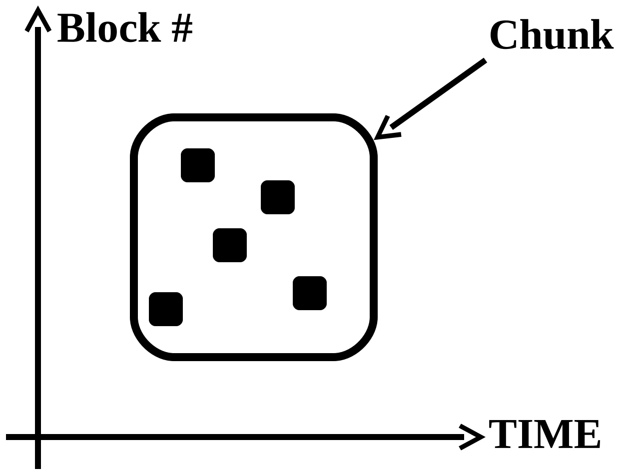
Speed up & Don't push too much



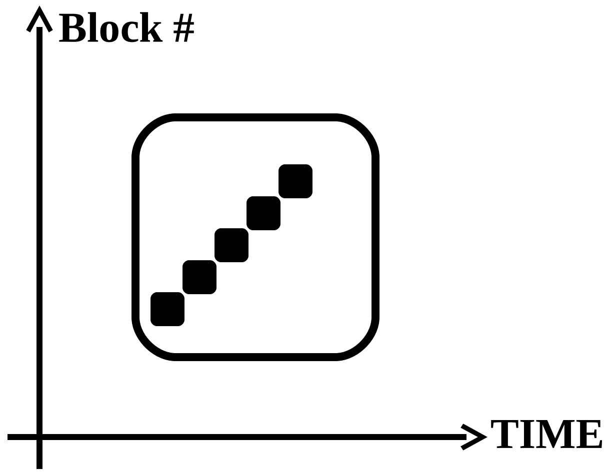
Accesses: Random vs. Sequential

- Disk arrays perform better with sequential accesses
  - A result of read-ahead (prefetching) policy
- Let disk arrays take care of sequential runs→ How?
  - Migrate random-accessed hot chunks into SSD
  - Don't increment access count for sequential accesses

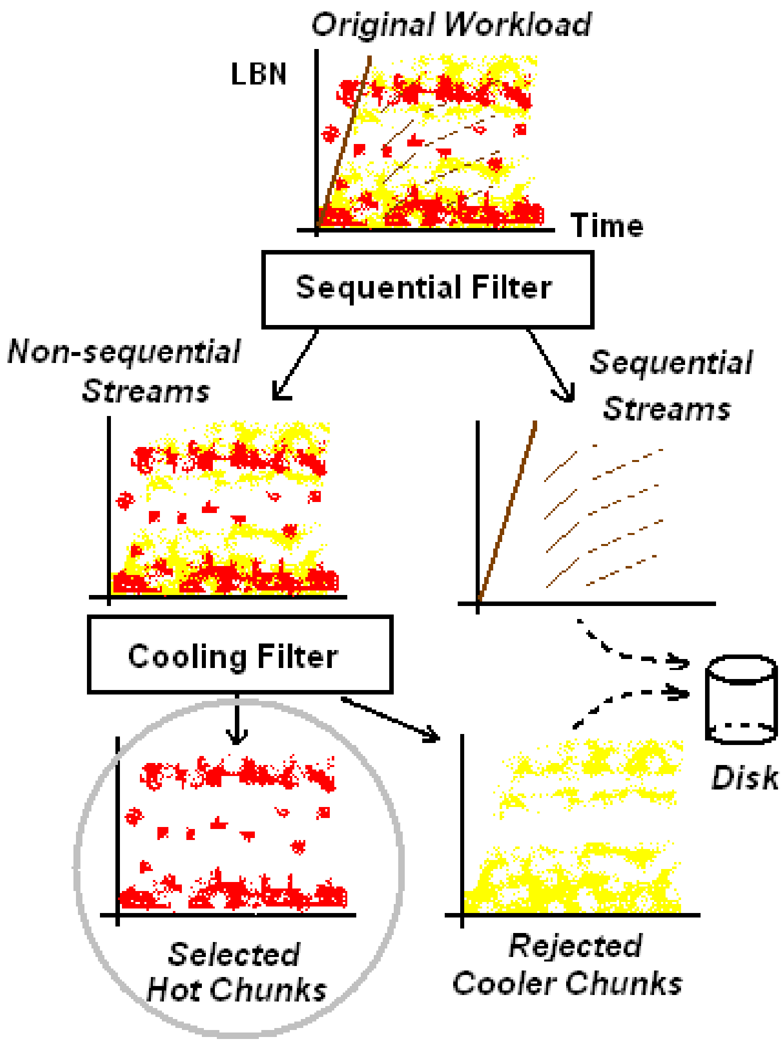
RANDOM

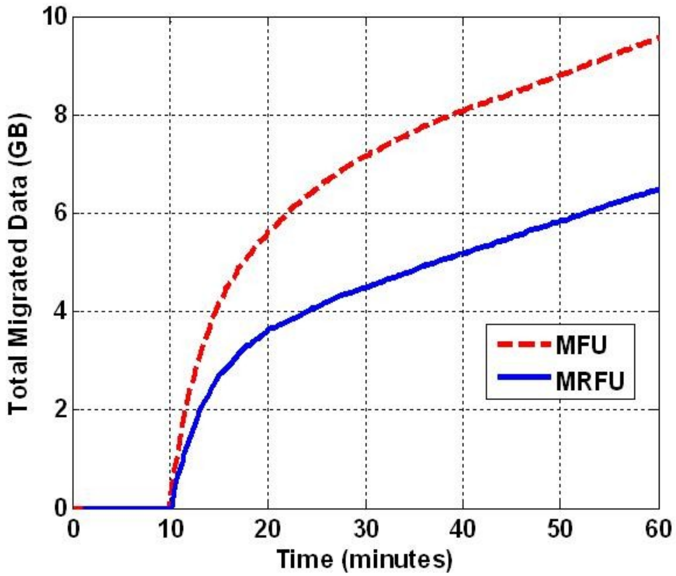


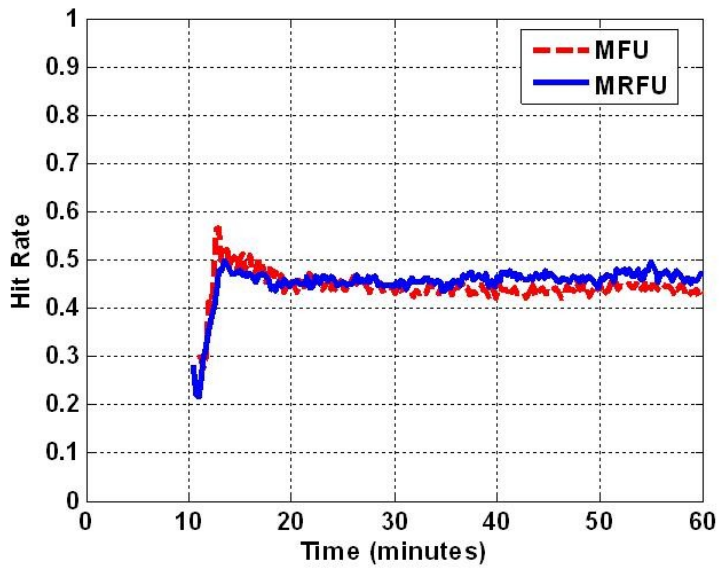
SEQUENTIAL



Workload Filters: Sequential & Cooling



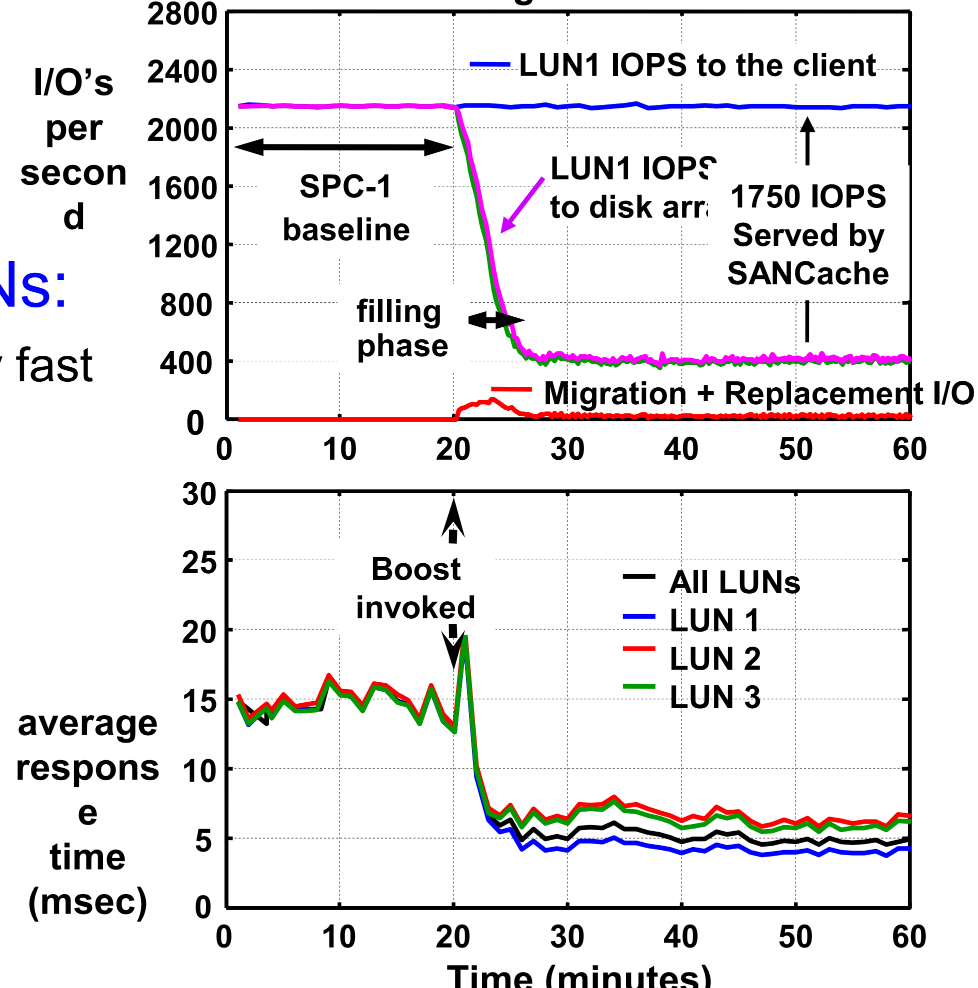




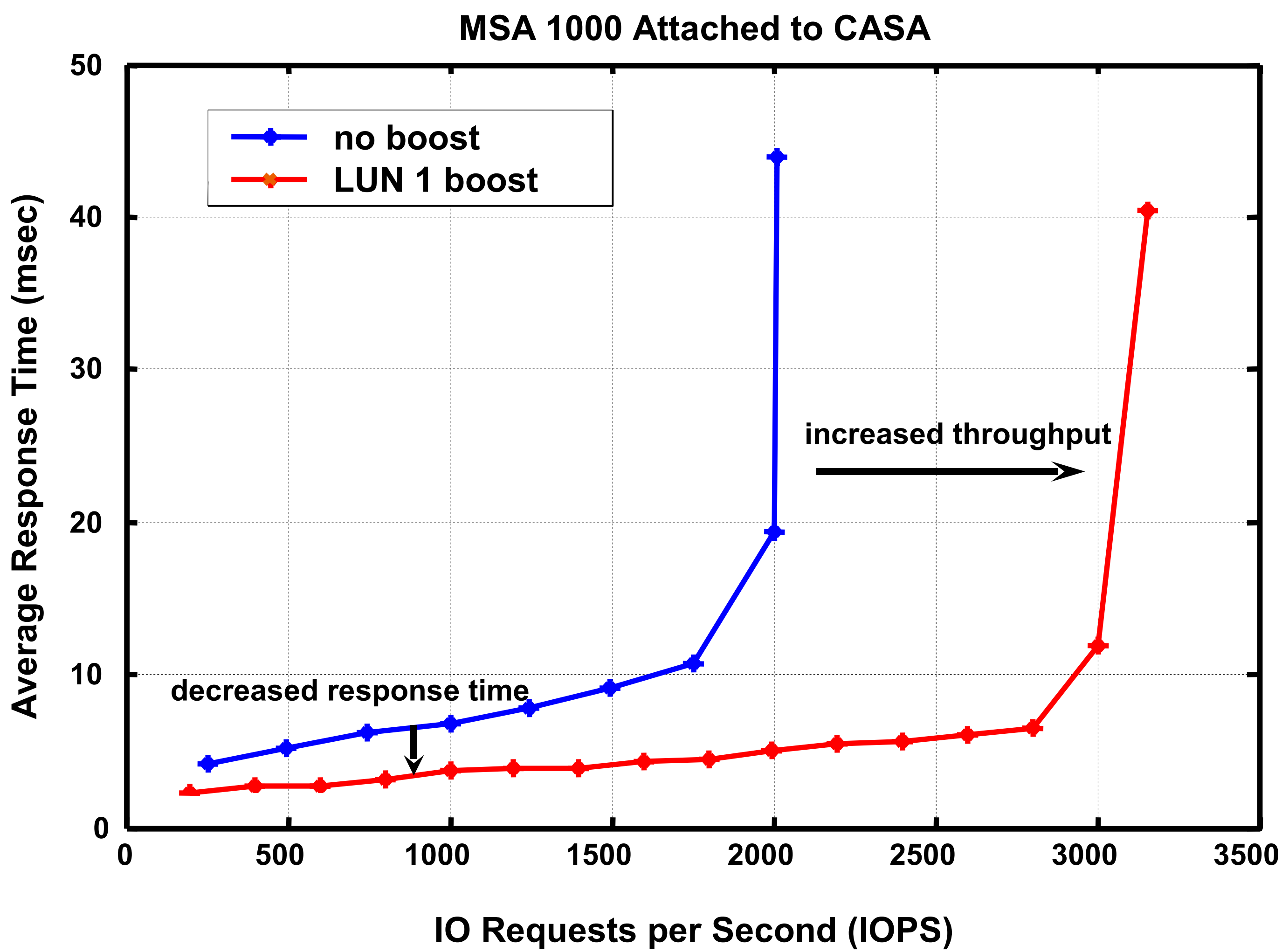
SANCache Performance Results

- Improves response time & throughput to clients
- For SANCache-enabled LUNs:
  - I/O's in "hot" regions are served by fast solid-state storage
- All LUNs benefit from decreased I/O load to disk arrays:
  - Fewer random I/O's to disk
  - Longer sequential runs to disk

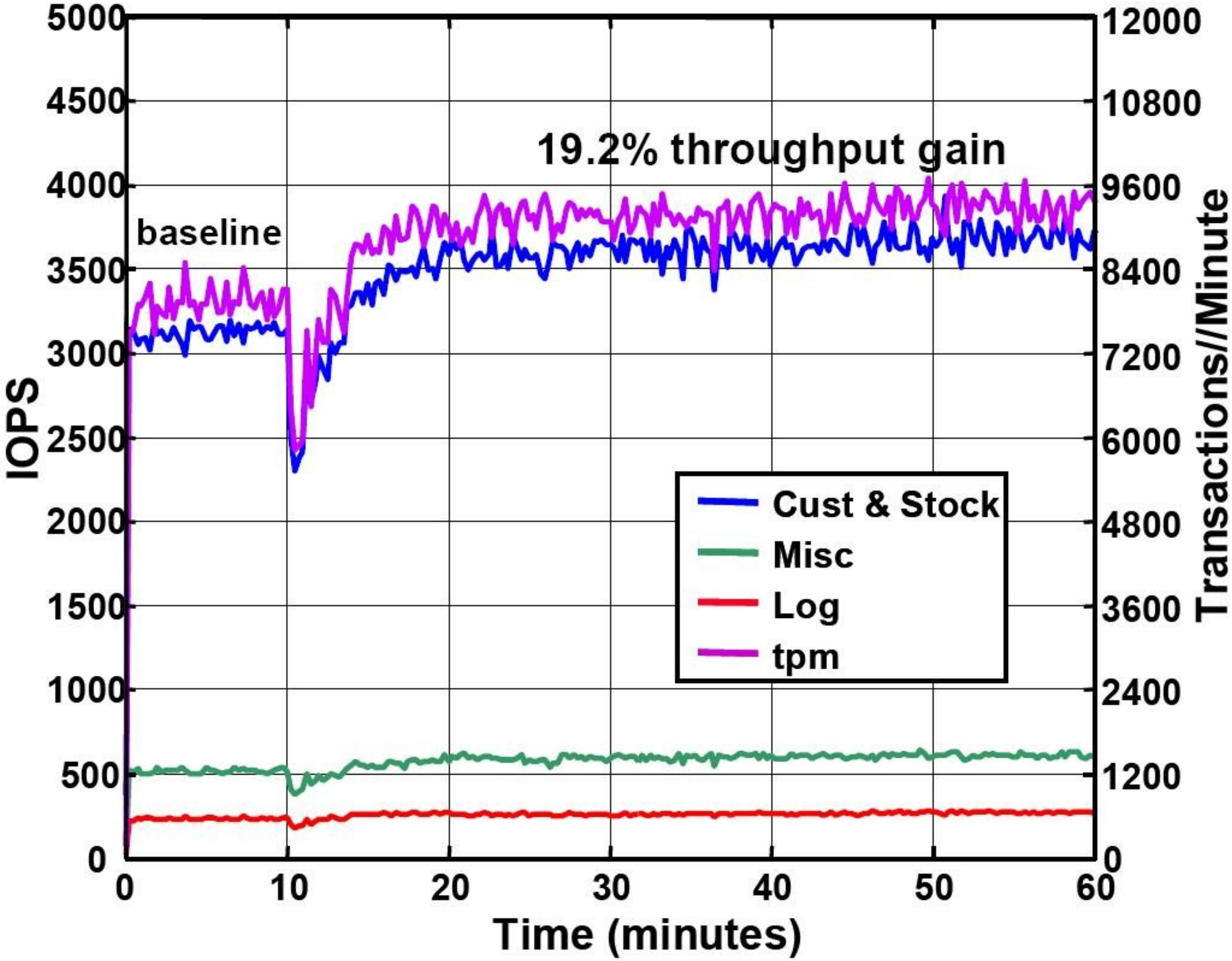
Sample results for boosting most active LUN in the SPC-1 storage benchmark workload



Performance Results (SPC1 on MSA)



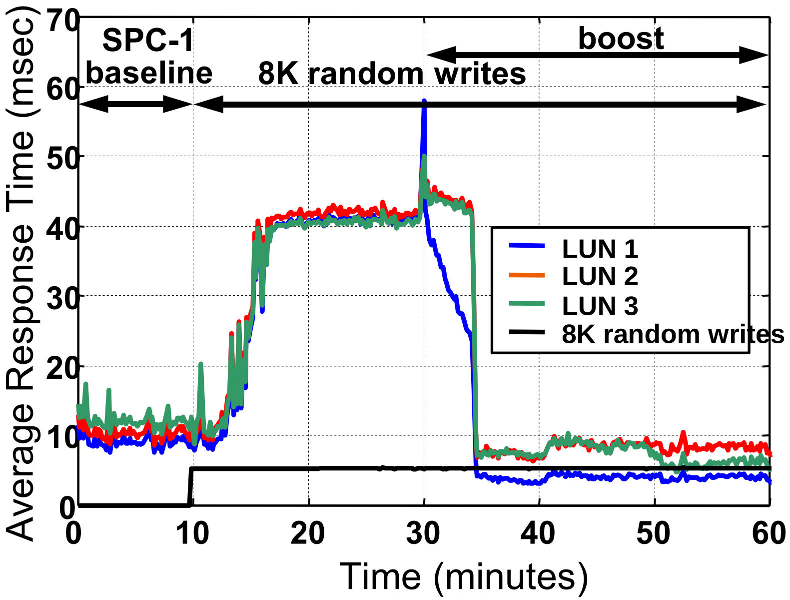
Performance Results (TPCC on XP1024)



SANCache Isolation Results

- While SPC-1 workload is operating on an XP1024 array, a write intensive workload is introduced
- The write workload is on a different LUN, port, & set of disk groups, but it shares the array's 32 GB of cache with SPC-1.

Alleviating Cache Interference



- The cache fills, causing the array to flush to disk with limited IOPS. This builds queue length, adversely impacting response time.
- Boosting SPC-1 LUN1 with 4 GB of SANCache absorbs I/O load from the XP array, restoring acceptable response time to SPC-1.

Ongoing and Future Work

- Testing SANCache over decentralized block storage (e.g. FAB)
  - SANCache migrates lots of chunks causing storage reconfigurations
  - Needs an efficient distributed metadata management algorithm
- QoSS issues and business aspects
  - Implemented API for clients to enter IOPS-latency goals (*i.e.* service contracts or utility functions)
  - Dynamic allocation of SANCache to maximize utility

Related Work

- Cache replacement algorithms
  - ARC-2Q, UBM-PCC
- Exclusive caching
  - Using heterogeneous algorithms or demotions
- Automated array configuration and data migration
  - Hippodrome, Aqueduct
- Web content distribution & placement
  - MFUPlace
- Quality of Storage Service
  - CacheCOW, Triage
- Disk array "cache partitioning"
  - RAMDisk or Cache LUN

Summary and Conclusions

- Addressed performance problems in SAN
  - Disk limitations and workload interference issues
  - A finer granularity, faster response, and higher accuracy method compared to manual configurations
- We prototyped SANCache and demonstrated…
  - Storage performance and workload isolation results
  - Using SPC-1, TPC-C loads on MSA & XP1024 arrays
- We believe that a balanced storage system design can only be achieved through *automation & adaptation* in today's complex SANs.