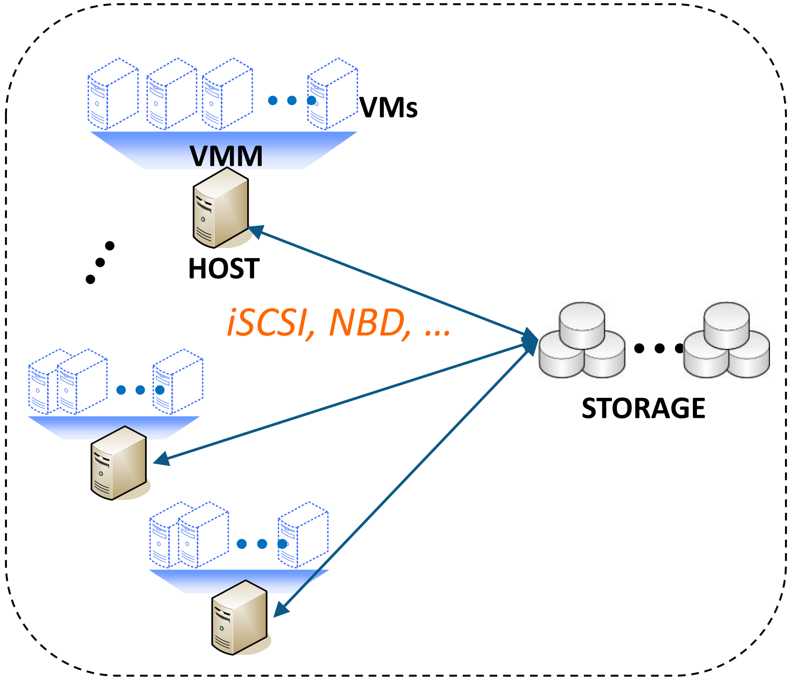
CloudCache: On-demand Flash Cache Management for Cloud Computing

Dulcardo Arteaga	Florida International University
Jorge Cabrera	Florida International University
Jing Xu	VMware Inc.
Swaminathan Sundararaman	Parallel Machines
Ming Zhao	Arizona State University

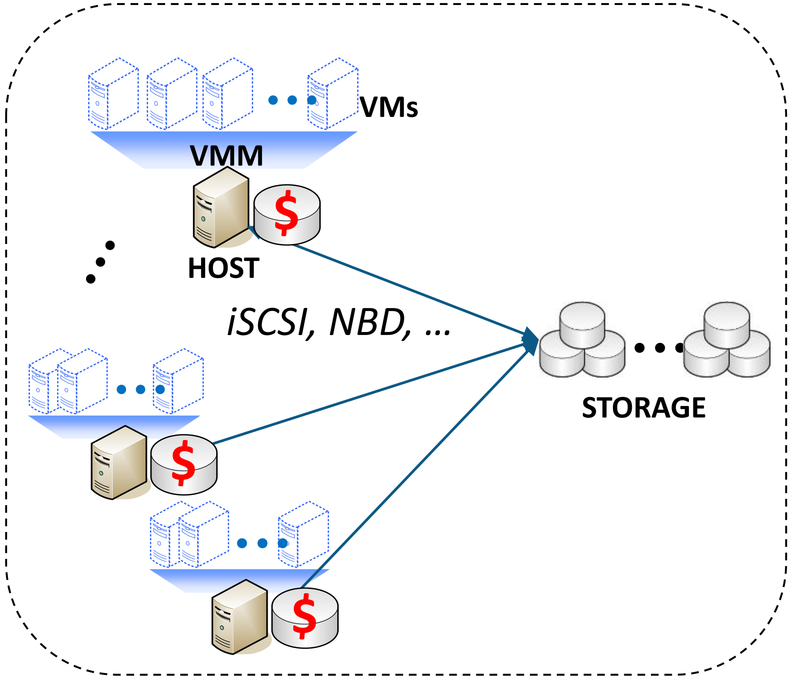
Background

- Networked storage systems are important building blocks for cloud computing
- Benefits
  - Efficient storage utilization
  - Reliable VM storage
  - Live VM migrations
- Challenge—scalability
  - Increasingly level of consolidation
  - Increasing data-intensive workloads



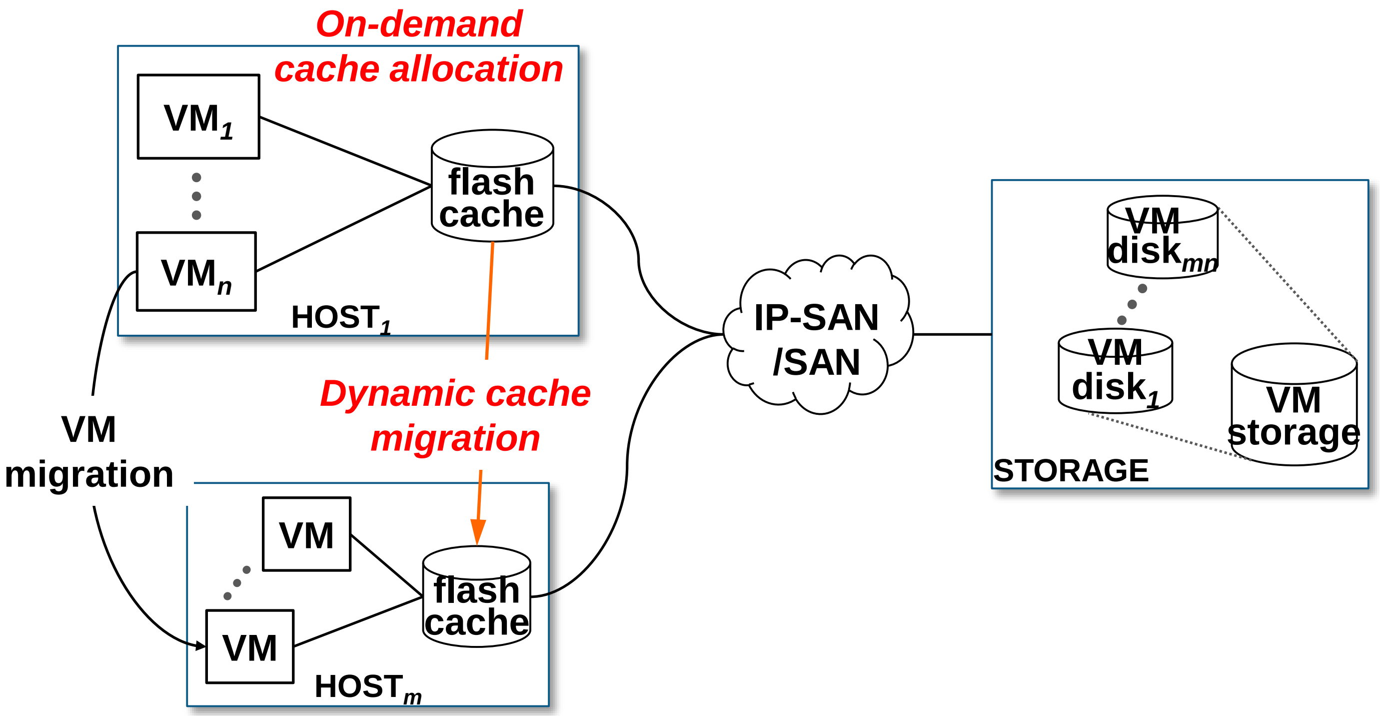
Flash caching to the rescue?

- Client-side caching
  - Exploit the locality in VM I/Os using the storage available on the client-side
- Flash-based cache devices
  - Exploit the high performance of flash storage
  - Avoid the long latency from the networked storage
- Challenges:
  - Limited cache capacity
    - Still small compared to dataset sizes
  - Limited device endurance
    - Caching makes it worse—both writes in the workloads and read misses cause wear-out
- Also applicable to other NVM based caches



Overview of CloudCache

- On-demand cache allocation
  - Allocate shared cache capacity to VMs according to their demands
- Dynamic cache migration
  - Balance cache load across hosts by migrating VMs and their cached data

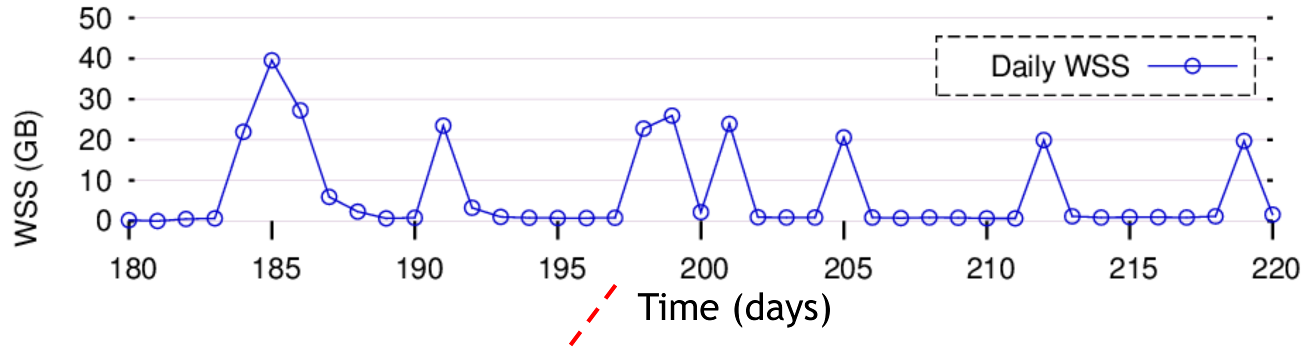


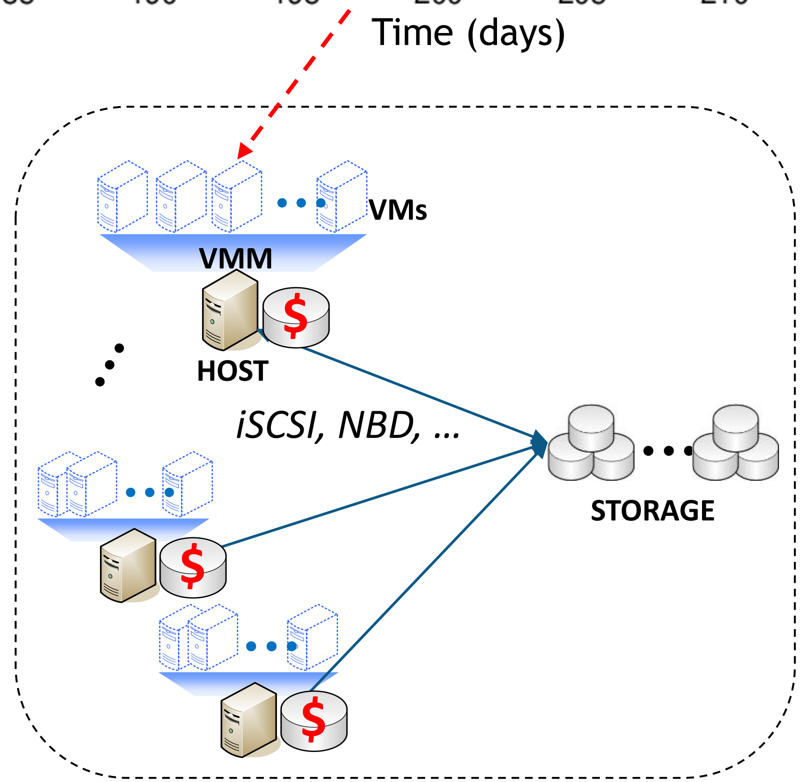
Outline

- Background
- **On-demand cache allocation**
- Dynamic cache migration
- Putting everything together

On-demand cache allocation

- Allocate cache capacity according to the workload cache demand
- How to model the cache demand of a workload?
- How to use the model to manage the cache?

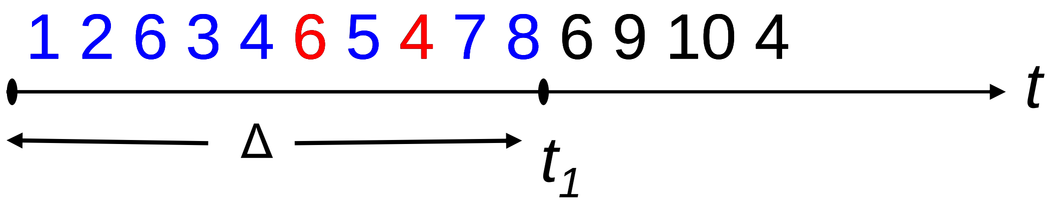




VISA

Working set and reuse working set

- Traditional Working Set *WS(t,T)* (Denning, 1968)
  - Set of distinct blocks referenced during [*t*-*T*, *t*]
  - Include data with low temporal locality
    - Waste cache space, hurt endurance
- Our proposed Reuse Working Set $RWS_N(t,T)$
  - Set of distinct blocks ***reused*** at least **N** times during [*t*-*T*, *t*]
  - Keep only the really useful data
    - Exclude low-temporal-locality data

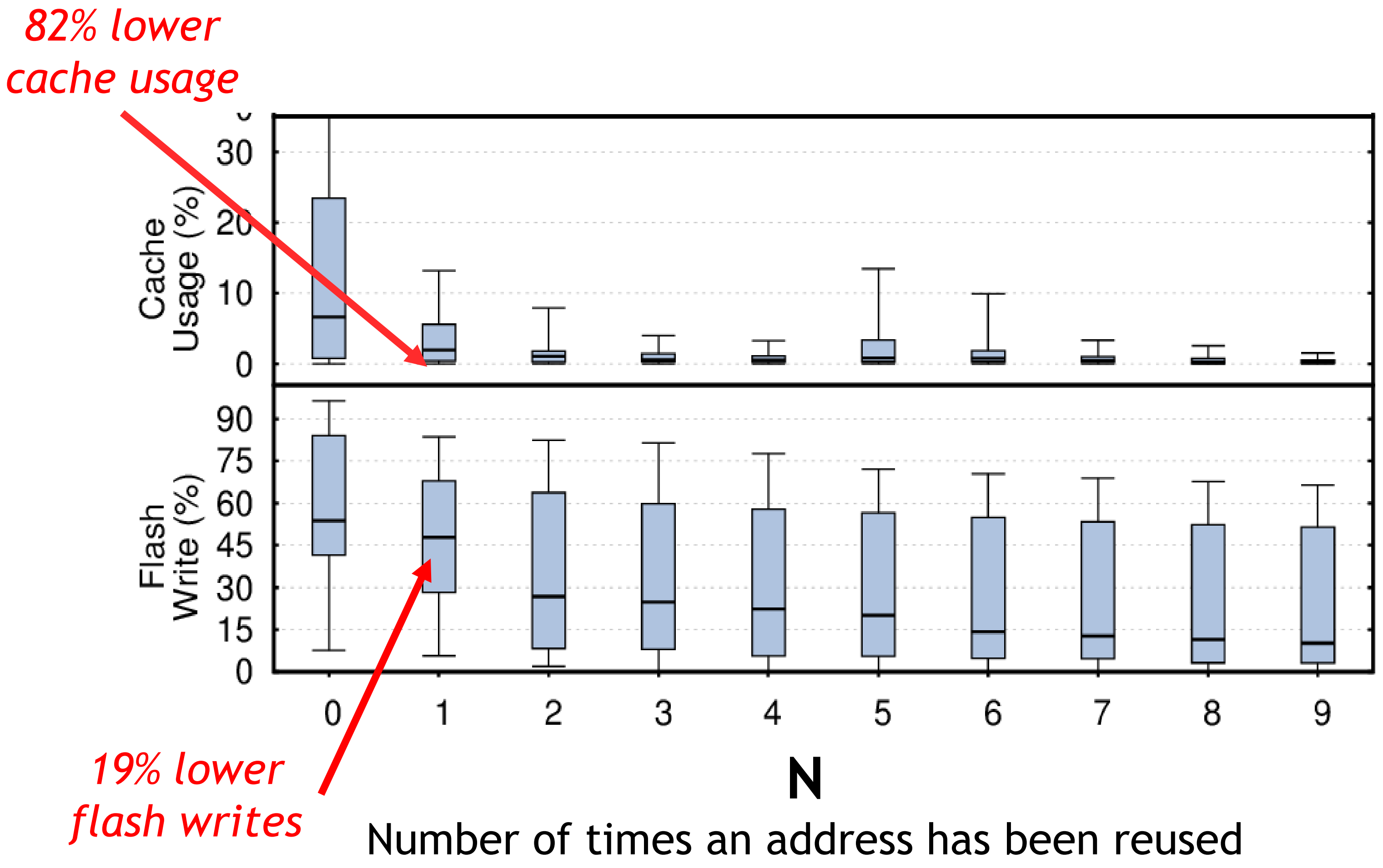


WS (Δ, t_1) = {1,2,6,3,4,5,7,8}   WSS = 8

RWS_1 (Δ, t_1) = {6,4} $RWSS_1$= 2

WS vs. RWS

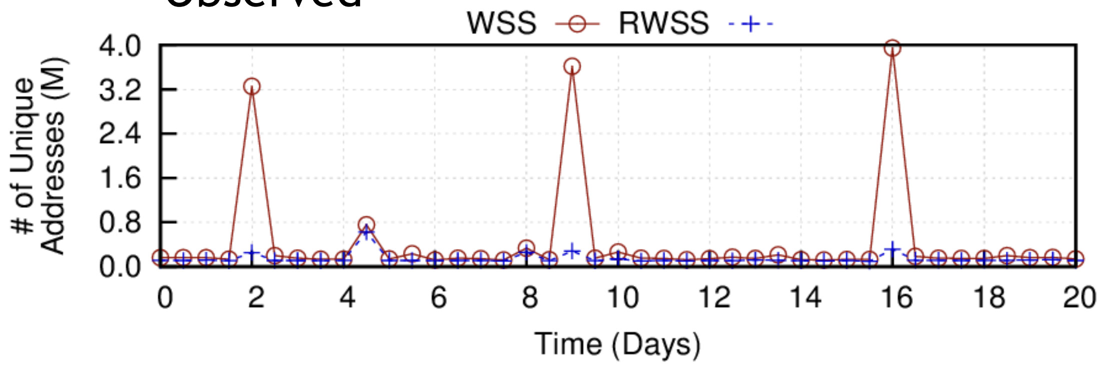
- Analysis of different *RWS*$_N$
  - 36 MSR traces
- RWS$_N$ — N is the number of times an address has been reused
- Flash write ratio: percentage of writes sent to the flash device
  - Indirect measurement of wear-out



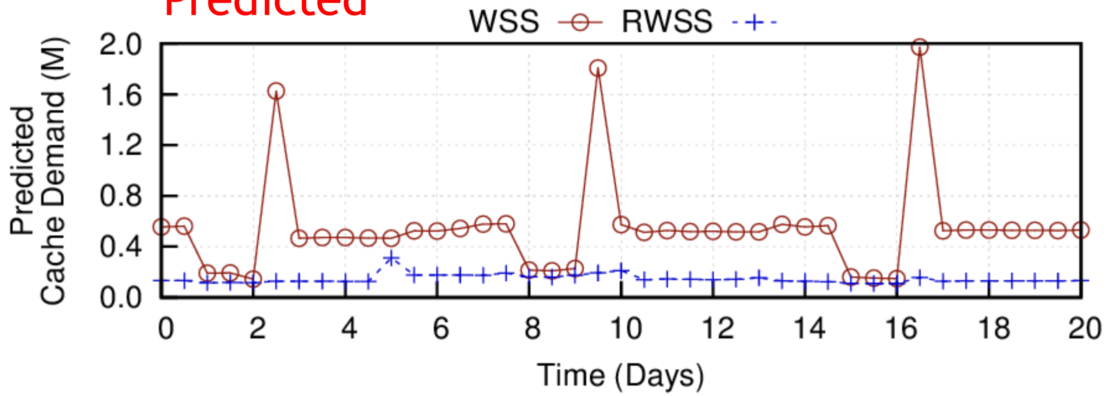
RWS-based cache allocation

- Measure the RWS size (RWSS) of each workload online
  - Window size typically set to days
- Predict the workload cache demands using observed RWSSes
  - Exponential smoothing with self-tuning smoothing factor
- Allocate cache capacity according to the predicted RWSSes
- Reduces cache usage up to 76%

Observed

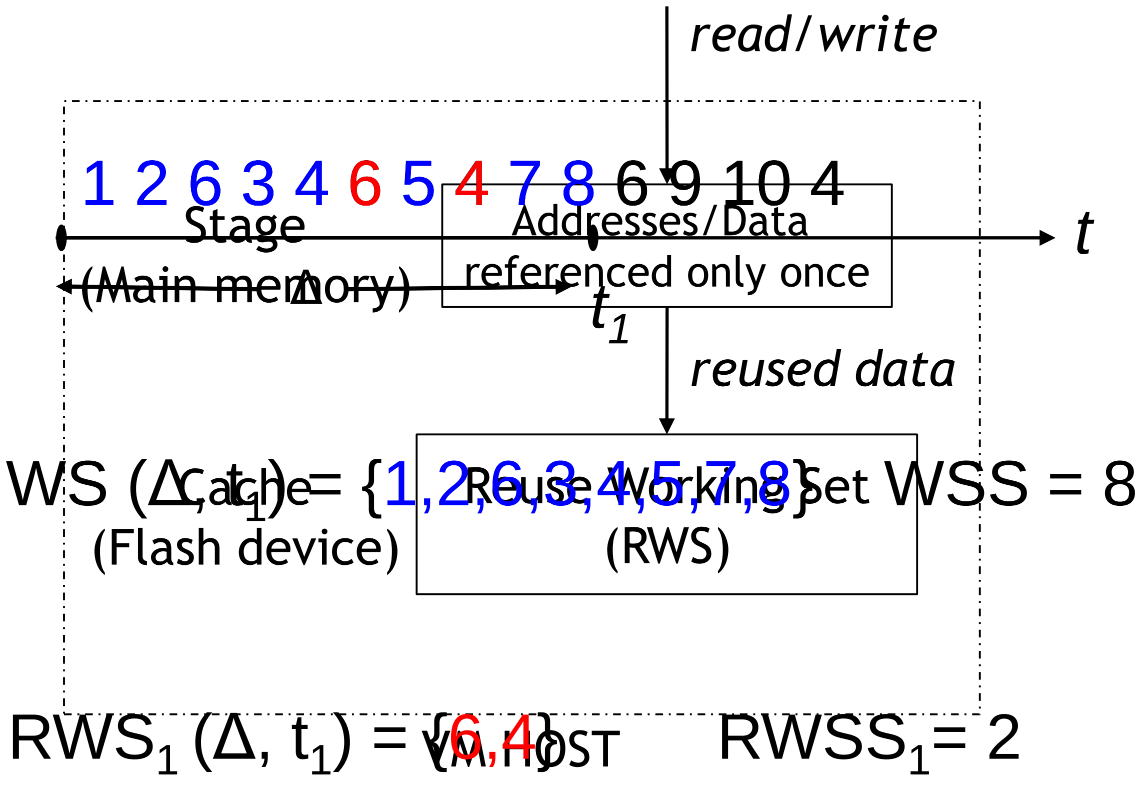


Predicted



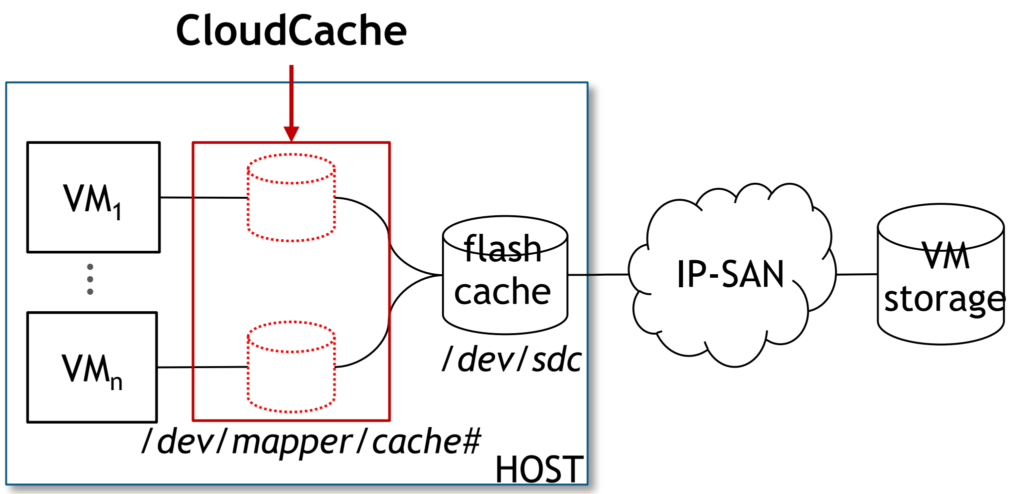
Cache admission

- Admit only reused data into cache
  - Avoid low-temporal-locality data from polluting the cache and causing unnecessary wear-out
- Staging—store candidates in memory before admitting them into cache
  - **Address staging**—stage only addresses of the candidates
  - **Data staging**—stage both addresses and data of the candidates
    - Reduce second-access misses
  - **Hybrid staging**—separate areas for staging addresses and data
    - Can stage more addresses than data items



Evaluation

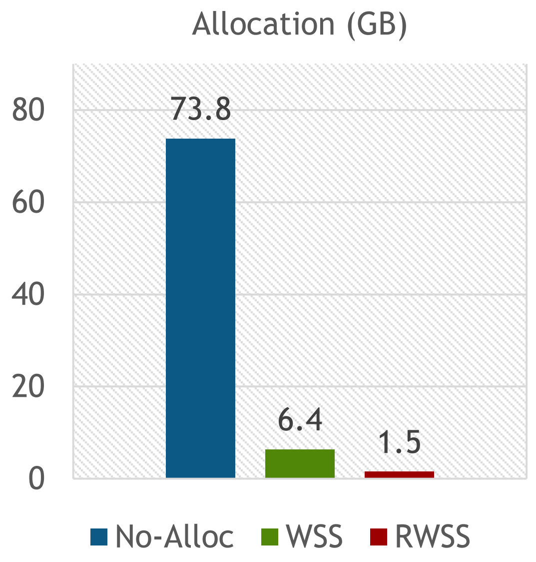
- Prototype
  - Based on block device virtualization (dm-cache, visa.lab.asu.edu/dmcache)
  - Transparent support for Linux-based (virtualized) environments
- Traces
  - Collected from several departmental servers
  - visa.lab.asu.edu/traces ([Systor'14])
- Testbed
  - iSCSI; Intel 120GB MLC SSD; XEN 4.1

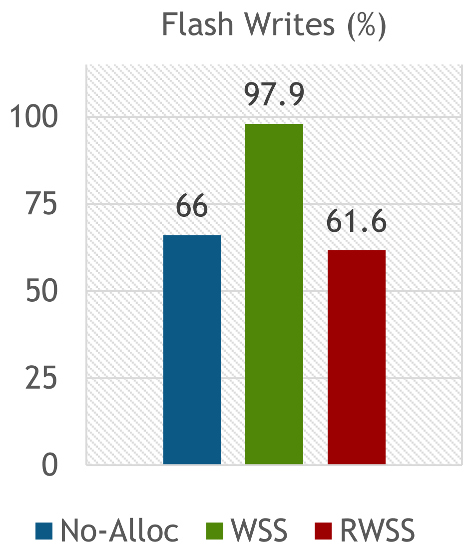


Name	Time (days)	Write (%)	WSS (GB)
Webserver	281	51	110
Moodle	161	13	226
Fileserver	152	22	1037

VISA

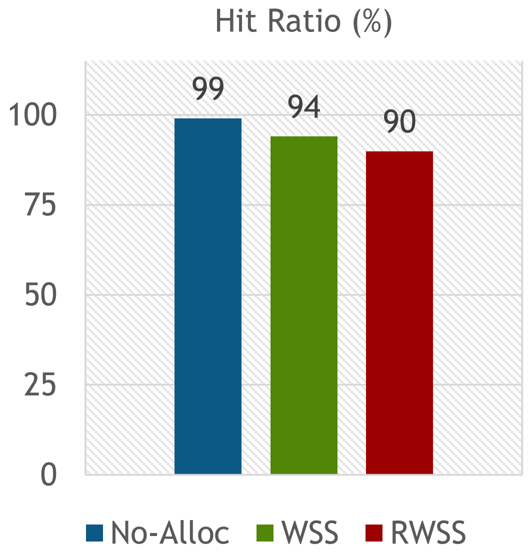
RWSS vs. WSS based allocation

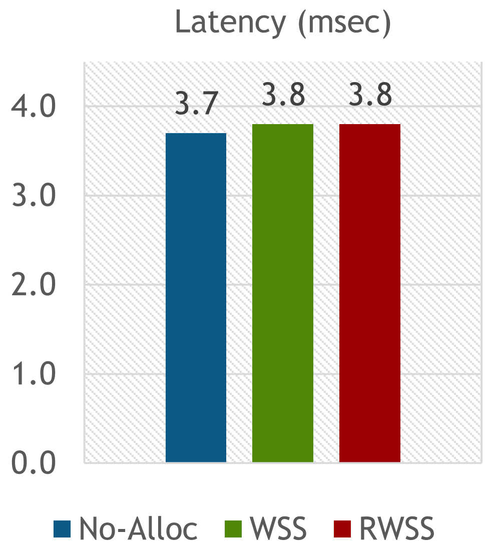




- Cache usage
  - Up to 72GB less than *No Allocation*
  - Up to 5GB less than *WSS*
- Flash write ratio
  - Up to 6% lower than *No Allocation*
  - Up to 37% lower than *WSS*

- Hit ratio
  - Up to 9% lower than *No Allocation*
  - Up to 4% lower than *WSS*
- Latency
  - 1% higher than *No Allocation*
  - Similar to *WSS*



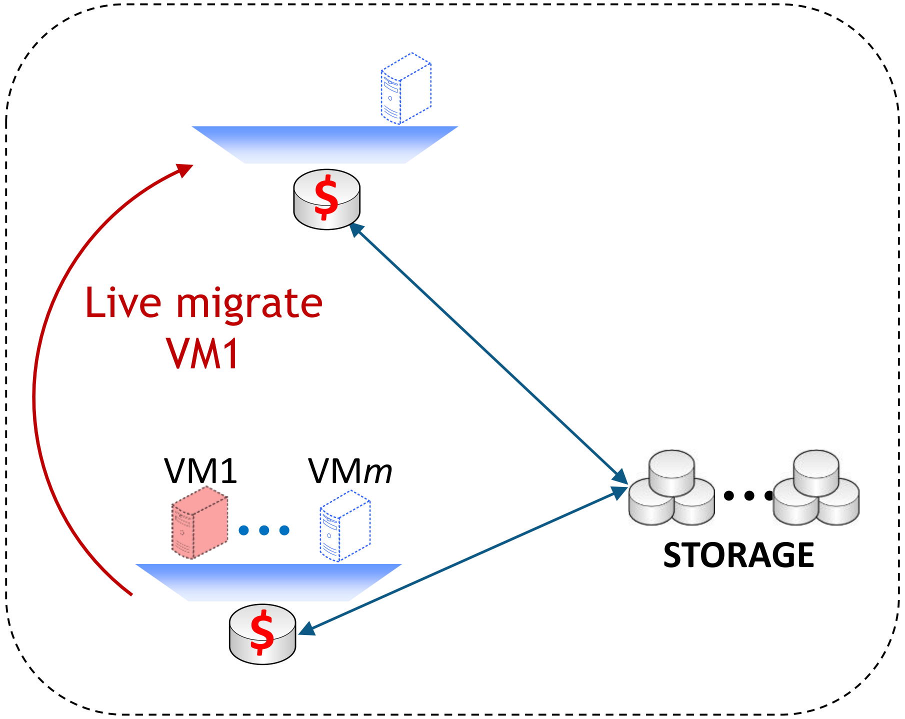


Outline

- Background
- On-demand cache allocation
- **Dynamic cache migration**
- Putting everything together

Dynamic cache migration

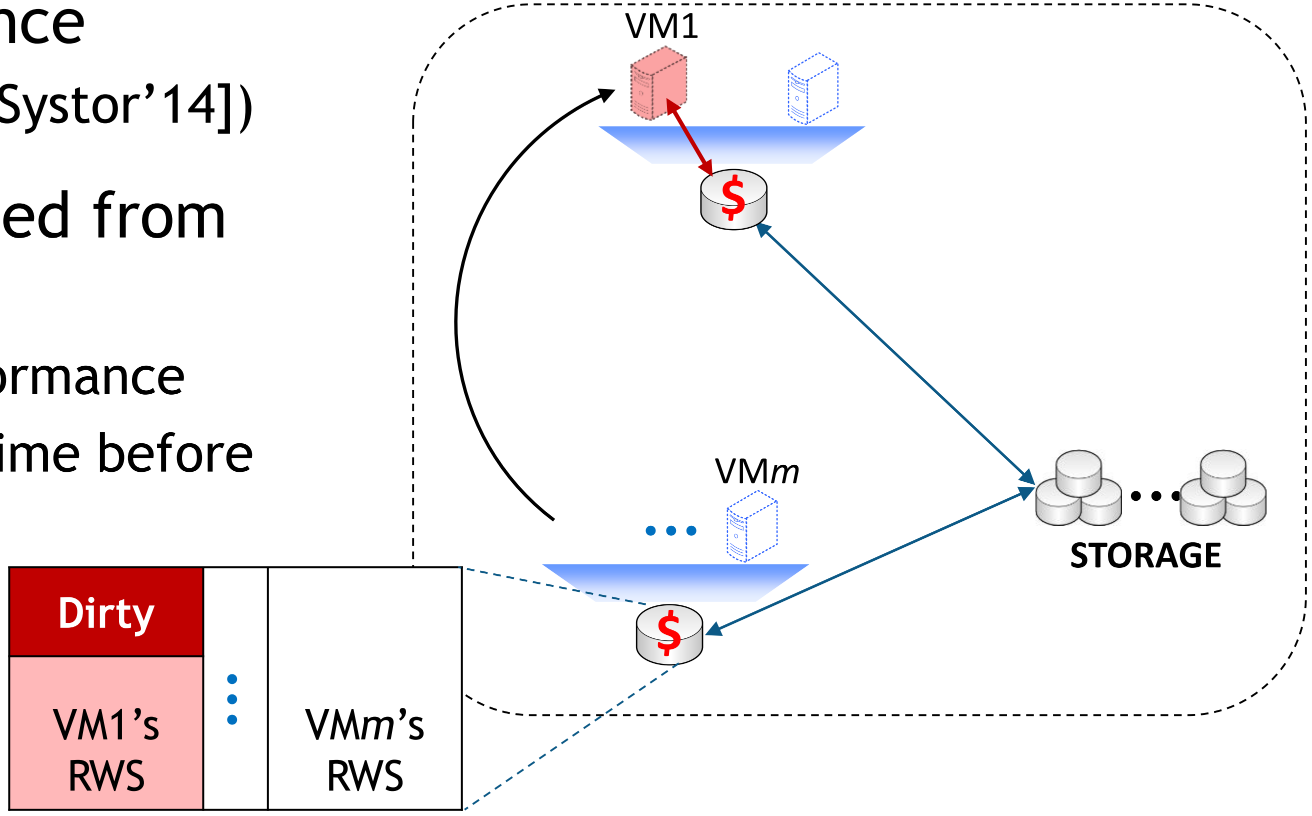
- CloudCache allocates cache according to VMs’ cache demands (RWSSes)
  - How to handle situations where the cache capacity is insufficient?
- Live VM migration can be used to balance the cache load across hosts
  - How to handle the cached (possibly dirty) data?



VISA

Challenges

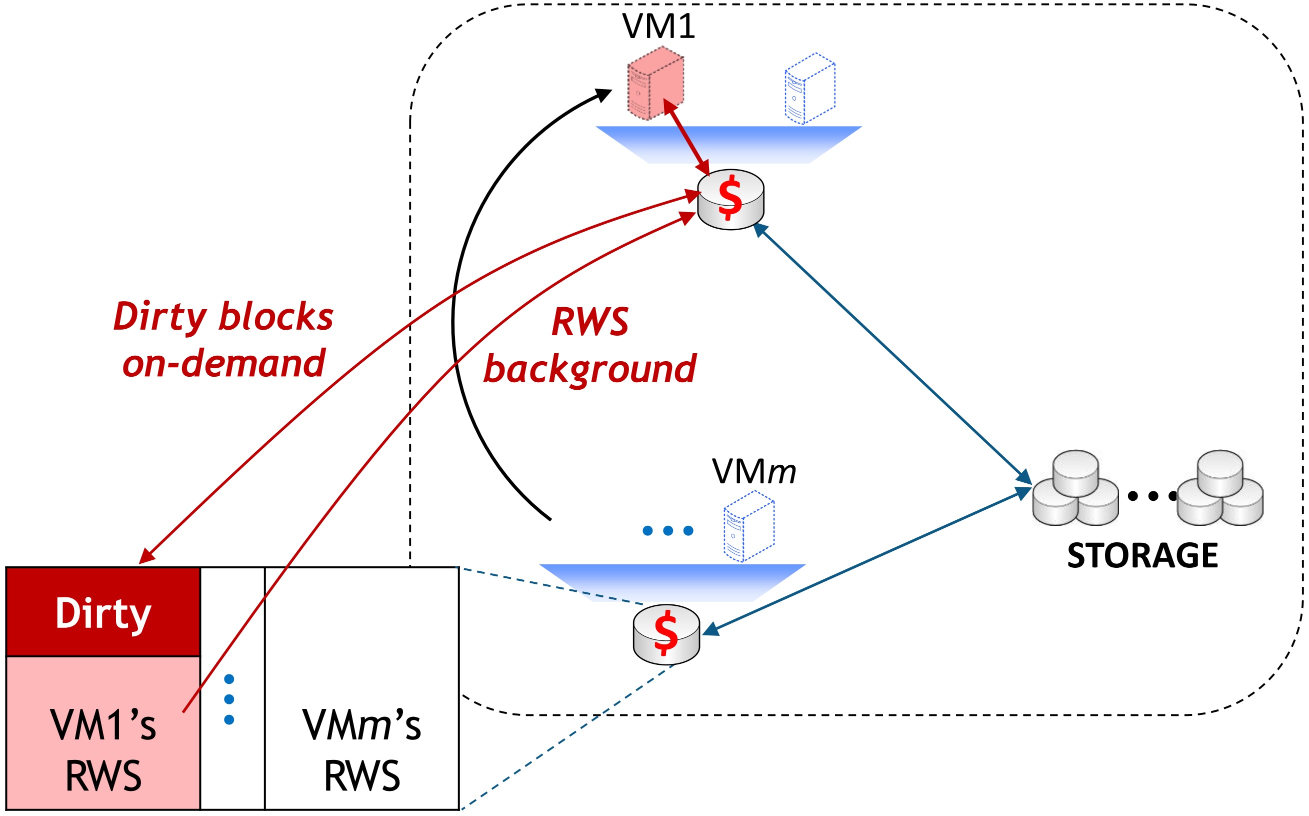
- Cached data is critical to performance
  - Warmup may take a long time ([ATC'13, Systor'14])
- Dirty pages on cache must be synched from source to destination
  - Write-back caching provides better performance
  - But flushing dirty data may take a long time before migration



VISA

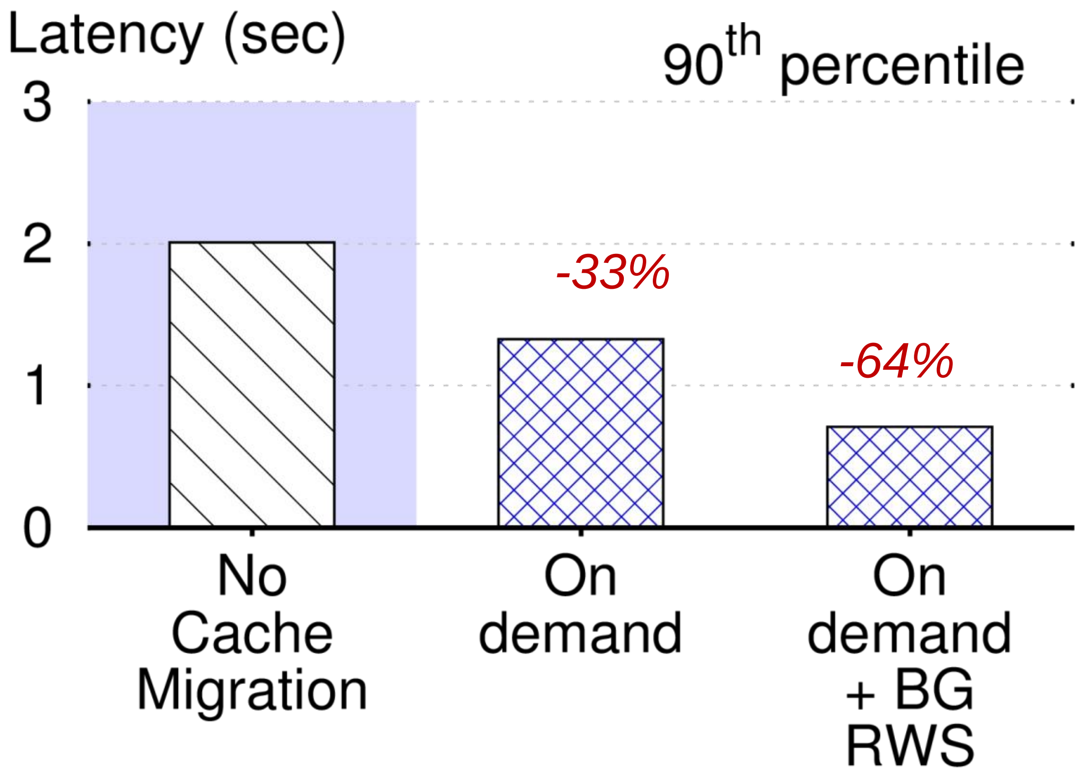
Dynamic cache migration

- **On-demand migration** of dirty data
  - Zero downtime to migrated VM
  - Cache immediately available to the VM
- **Background migration** of RWS
  - Quickly warmup the cache
  - Migrate only the useful data
- Rate limiting
  - Limit number of blocks transferred per period of time
  - Limit the impact to co-hosted VMs



Evaluation

- A day-long moodle trace
  - Read intensive
  - RWS: 5GB, 15% dirty
- On-demand migration enables zero downtime
  - 54s downtime otherwise
- Background migration allows fast cache warm-up



Migrated VM's latency

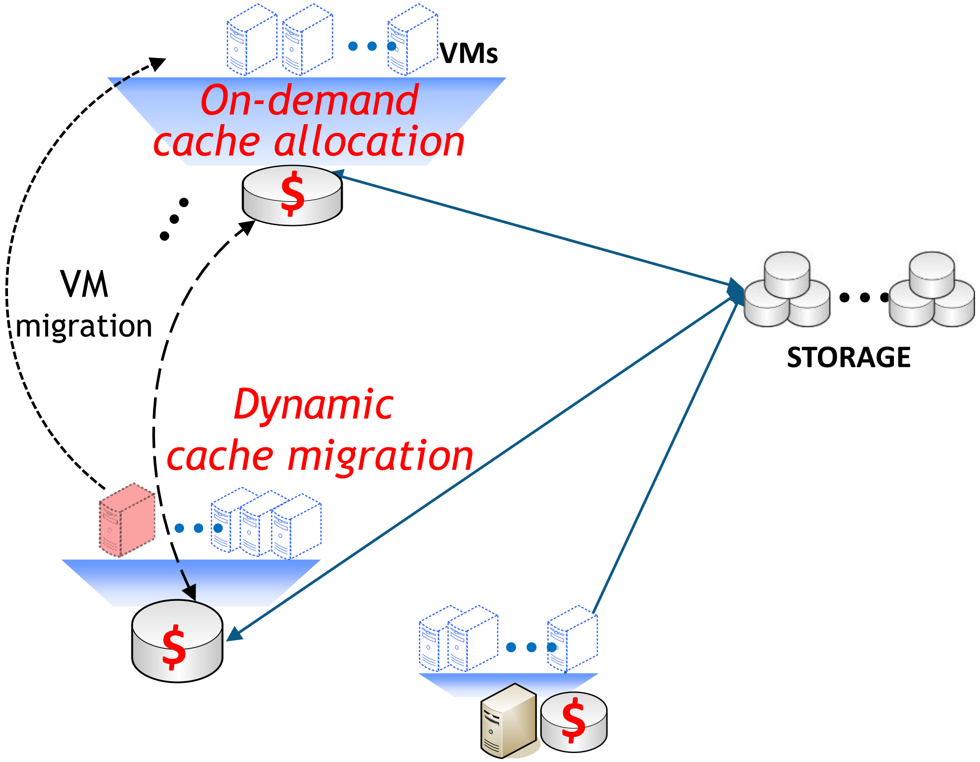
Outline

- Background
- On-demand cache allocation
- Dynamic cache migration
- **Putting everything together**

VISA

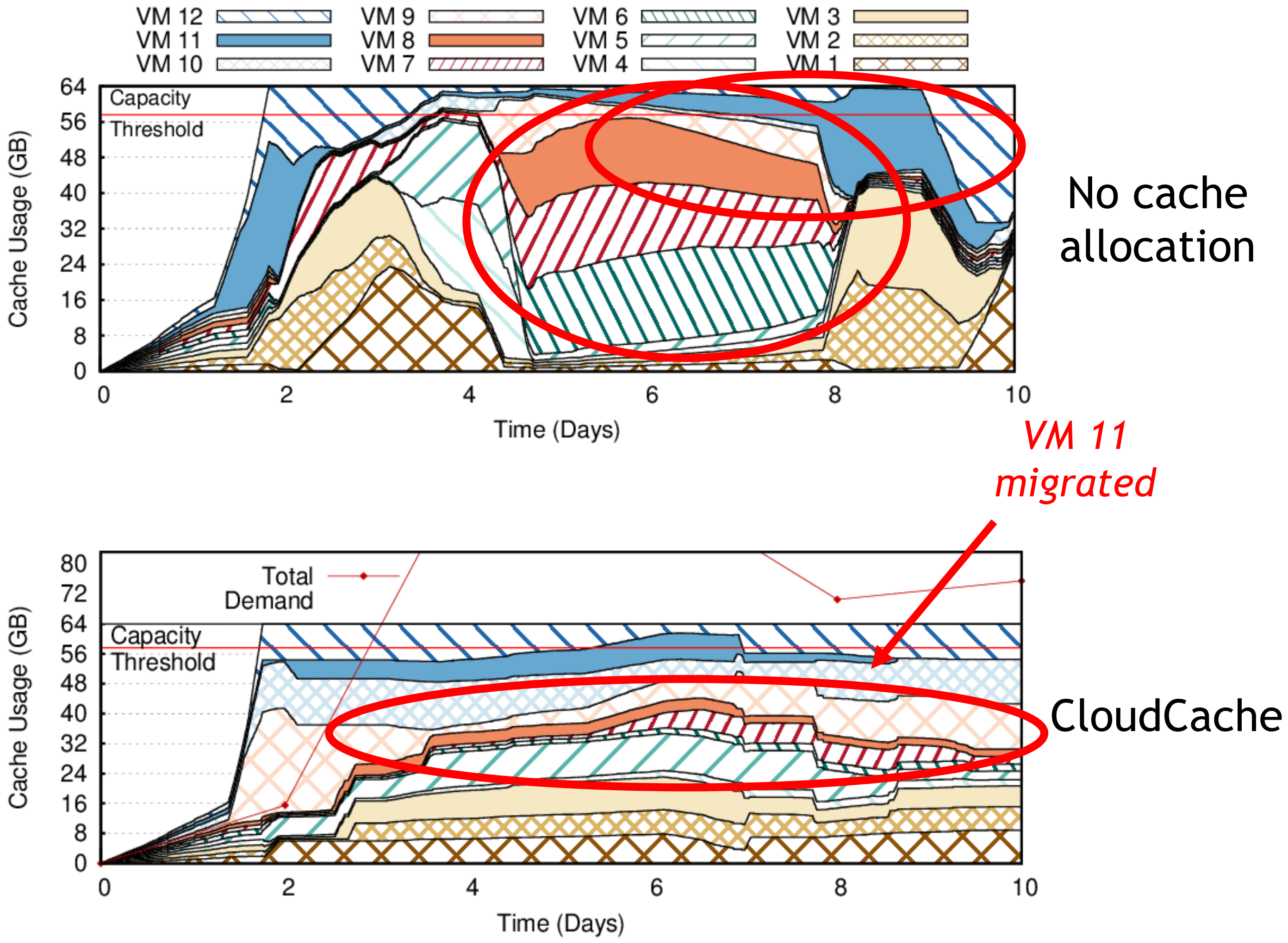
Putting everything together

- Allocate cache capacity proportionally to RWSSes
- Migrate VMs and their cached data when a cache is overloaded
  - After exceeding the 90% watermark for 3 consecutive periods
- Choose the destination host by minimizing the cache load imbalance



Evaluation

- Setup
  - 2 hosts each w/ 64GB cache
  - 10-day Webserver trace
  - 12 VMs
- RWS-based allocation allows every VM got a fair share
- Dynamic cache migration balances cache load
- 28% higher hit ratio, 27% lower a latency



VISA

Related work

- Cache allocation (S-CAVE, vCacheShare, Centaur)
  - Admit all referenced data into cache
  - Do not consider dynamic cache migration to deal with overloaded cache
- Cache admission (HEC, LARC)
  - Do not consider how to allocate shared cache to concurrent workloads
  - RWSS-based cache admission achieves better reduction in cache footprint and wear-out
- Processor and memory cache allocation ([ICPADS’01, ASPLOS’04, ASPLOS’09, MICRO’08, FAST’03])
  - Flash cache presents different challenges and opportunities
  - Low-locality data are detrimental to performance and endurance

Conclusions

- The RWS model can capture data with good temporal locality
- Allocation based on RWSS can efficiently meet workload cache demands
- Dynamically migrating VMs and cached data can effectively balance cache load across host with minimal performance impact
- Our results show:
  - Single VM: Up to 76% reduction in cache usage and up to 37% in flash writes
  - 12 concurrent VMs: 28% higher hit ratio and 27% lower latency, on average

Acknowledgements

- Sponsors
  - National Science Foundation CAREER award CNS-125394
  - Department of Defense award W911NF-13-1-0157
- VISA research lab
  - http://visa.lab.asu.edu

- *Thank you!*