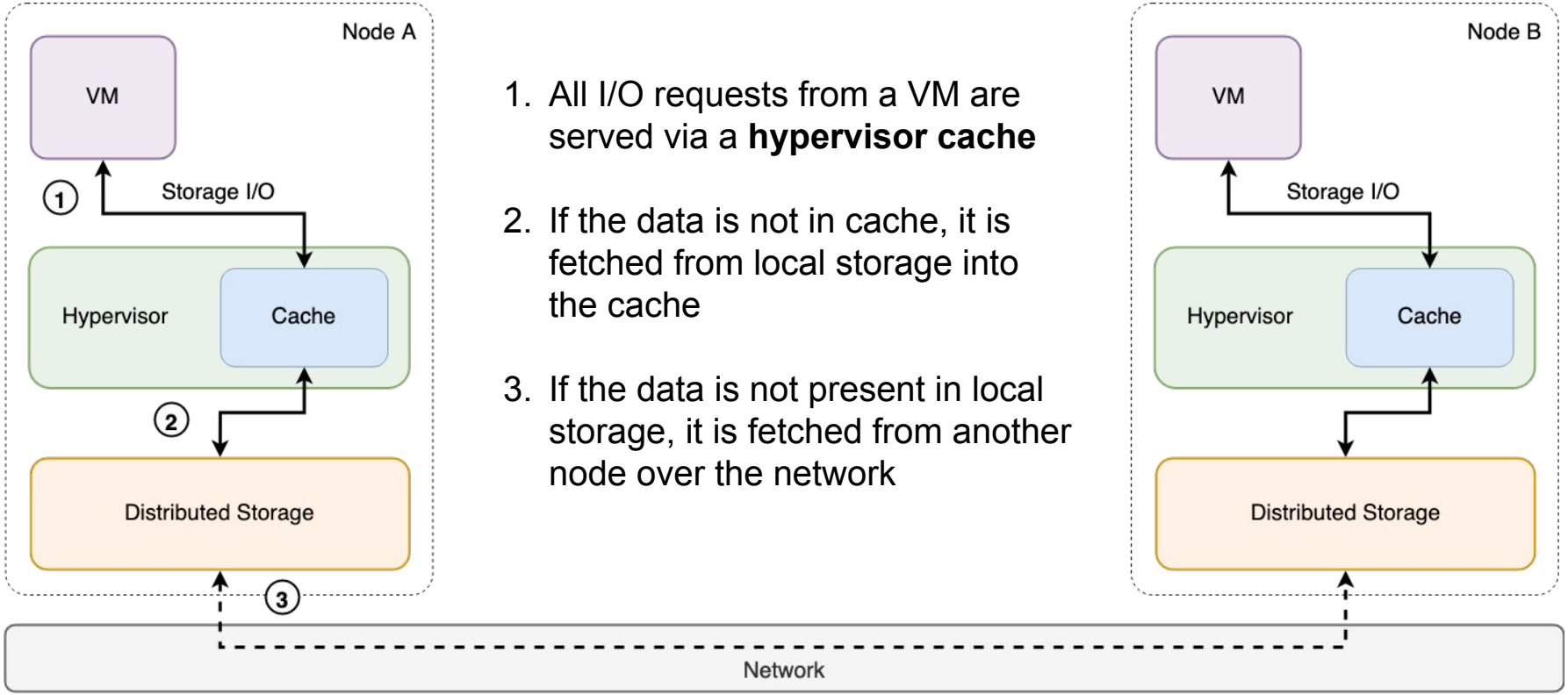
PRESTO: PREwarming STOrage Caches for Improving I/O Performance in Virtualized Infrastructure

presented by
Sukrit Bhatnagar (173059003)

guided by
Prof. Purushottam Kulkarni

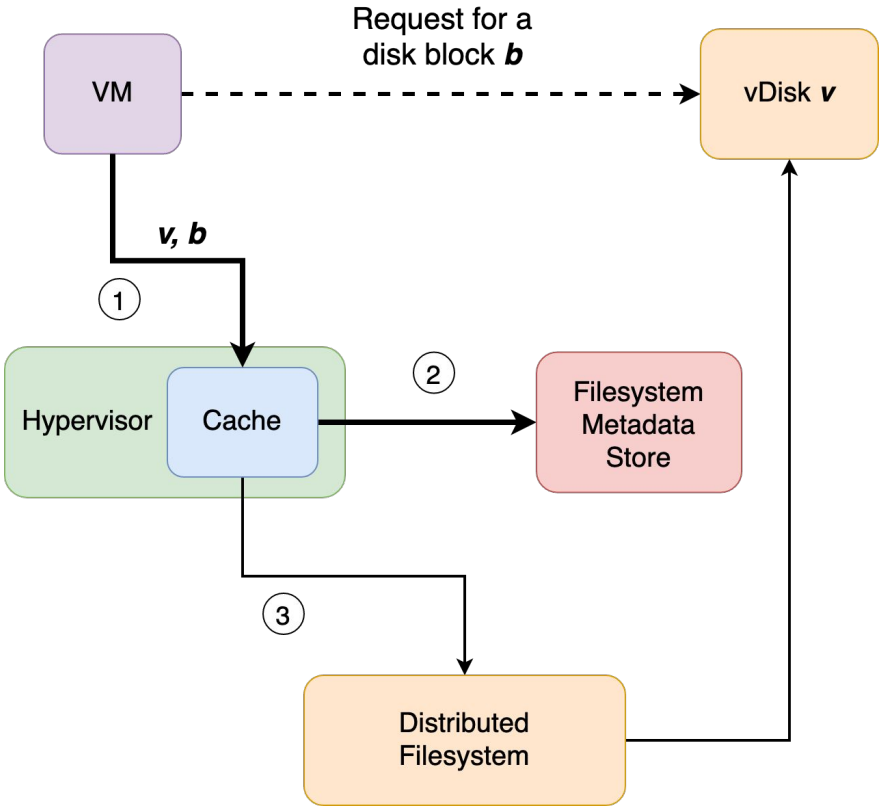
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

I/O Path with Distributed Storage



1. All I/O requests from a VM are served via a **hypervisor cache**
2. If the data is not in cache, it is fetched from local storage into the cache
3. If the data is not present in local storage, it is fetched from another node over the network

Closer look at the path...



1. VM makes an I/O request, intercepted by the cache

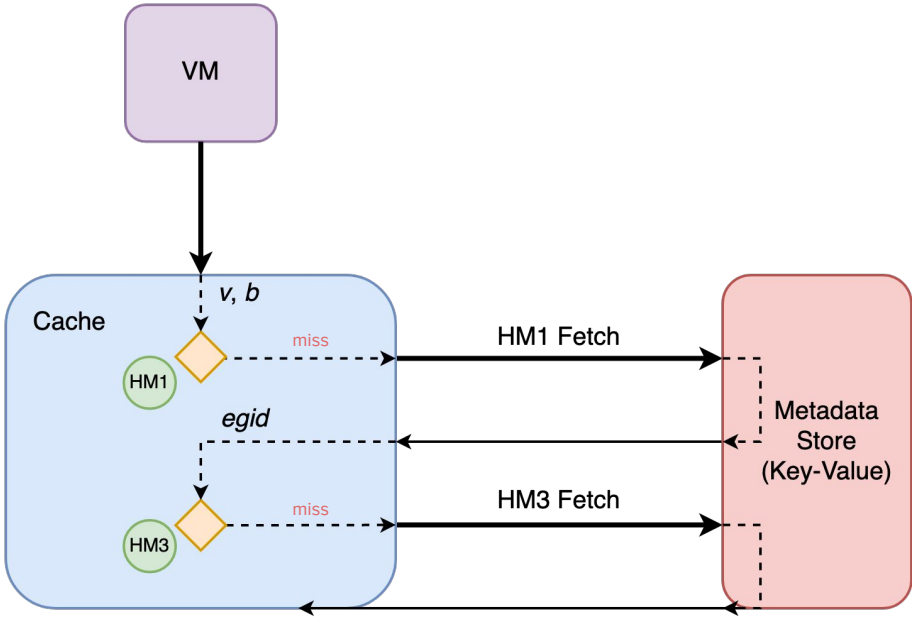
2. Metadata is looked up to locate the disk block data
   (and is cached thereafter...)

3. After translating metadata, cache serves the block data
   (and caches it as well...)

Inside the cache: objects

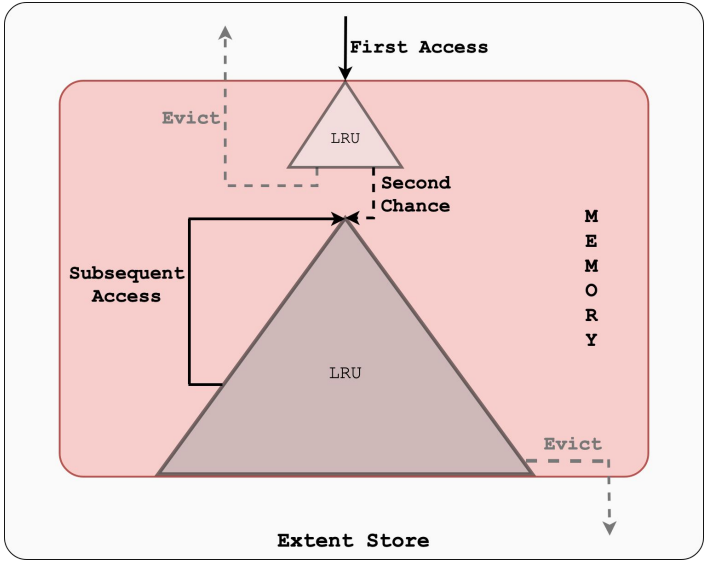
- caching fs metadata as objects
- two types (layers) of objects
  - HM1 & HM3

- get HM1 key from VM request
- locate disk block using HM3 val



HM = Hash Map, egid = egroup ID

Inside the cache: pools



- 2 pools in memory (Δ)
  - single-touch
  - multi-touch
- LRU replacement for both
- multi-touch size > single-touch size (80%-20% split)

Cache as an engine

Cold cache: empty cache or filled with irrelevant objects; ↓ Hit Ratio

Warm cache: cache filled with (temporally) relevant objects; ↑ Hit Ratio

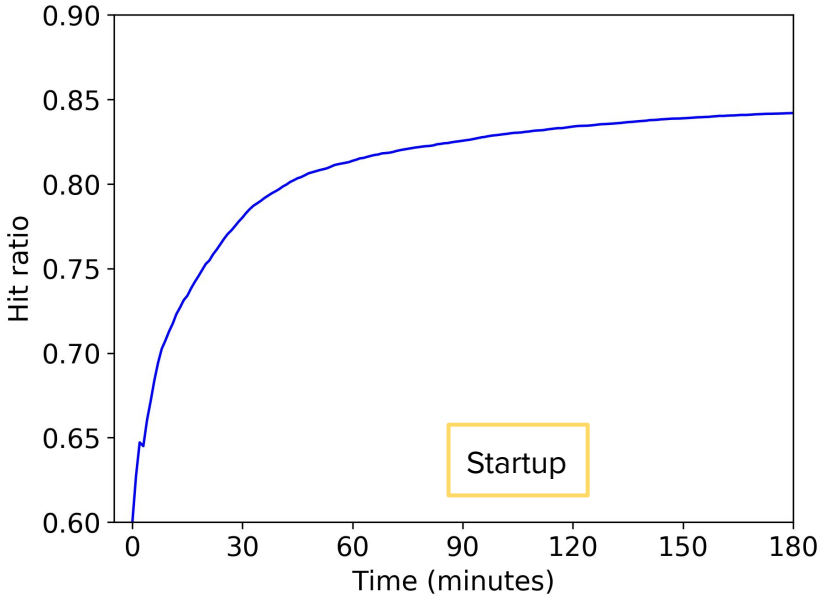
Cache as an engine

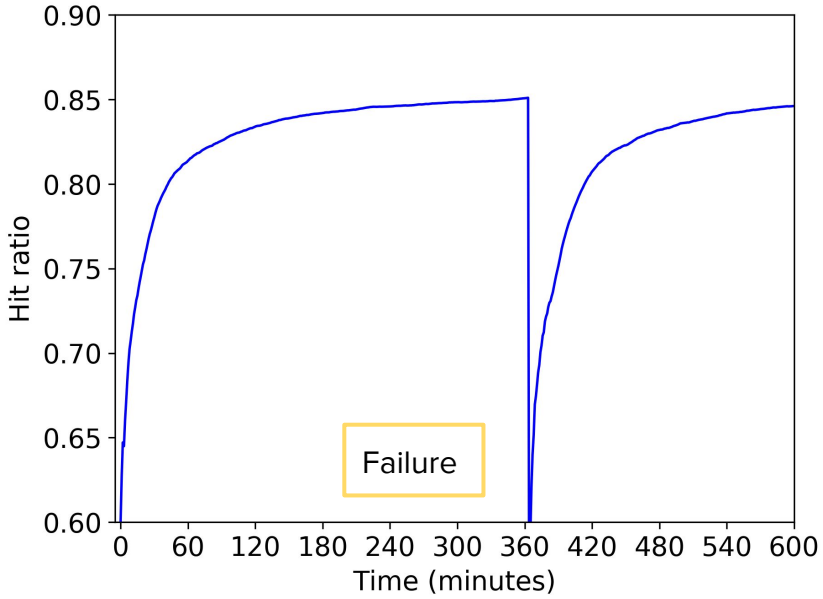
Cold cache: empty cache or filled with irrelevant objects; ↓ Hit Ratio

Warm cache: cache filled with (temporally) relevant objects; ↑ Hit Ratio

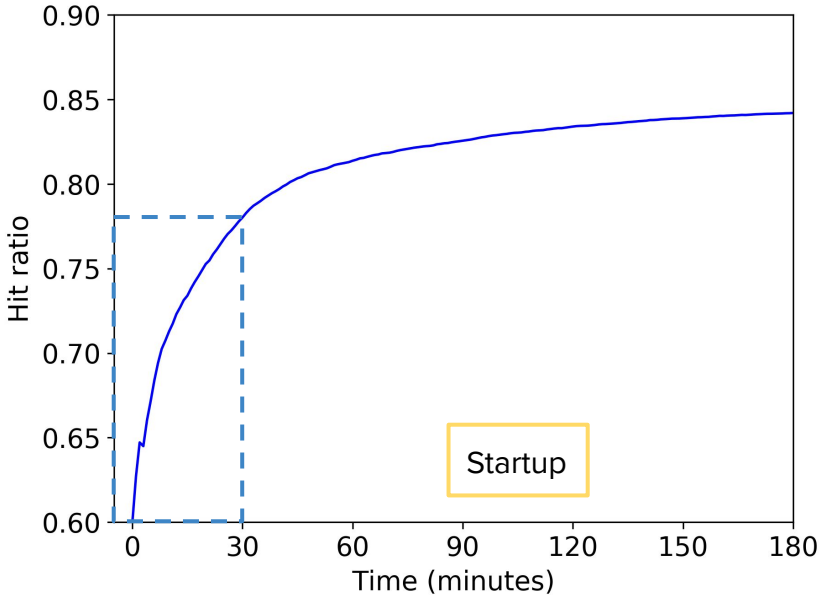
Prewarming: proactively load important objects into the cache

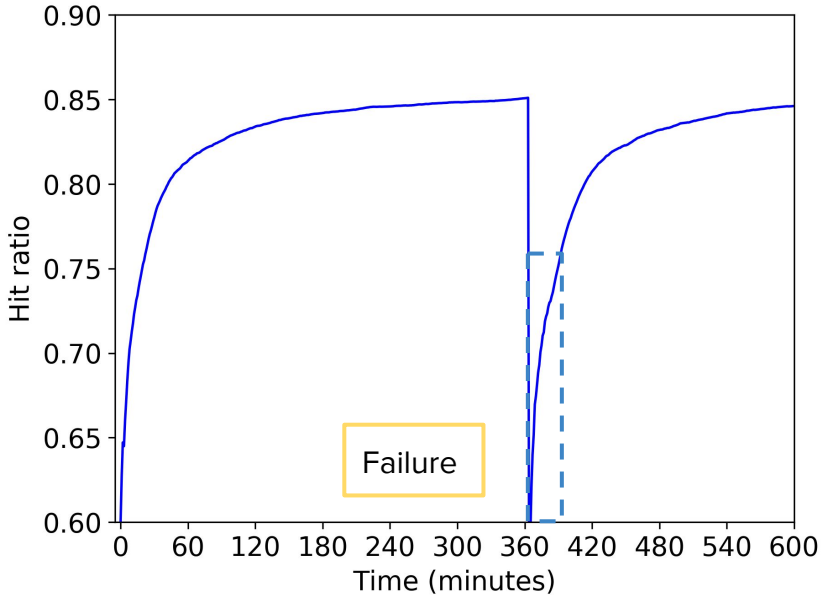
Scenarios for a cold cache





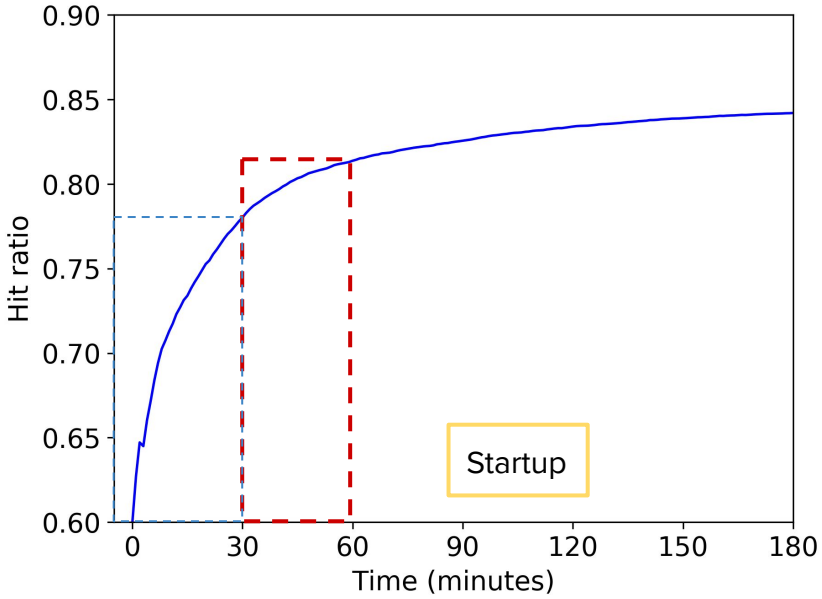
Scenarios for a cold cache

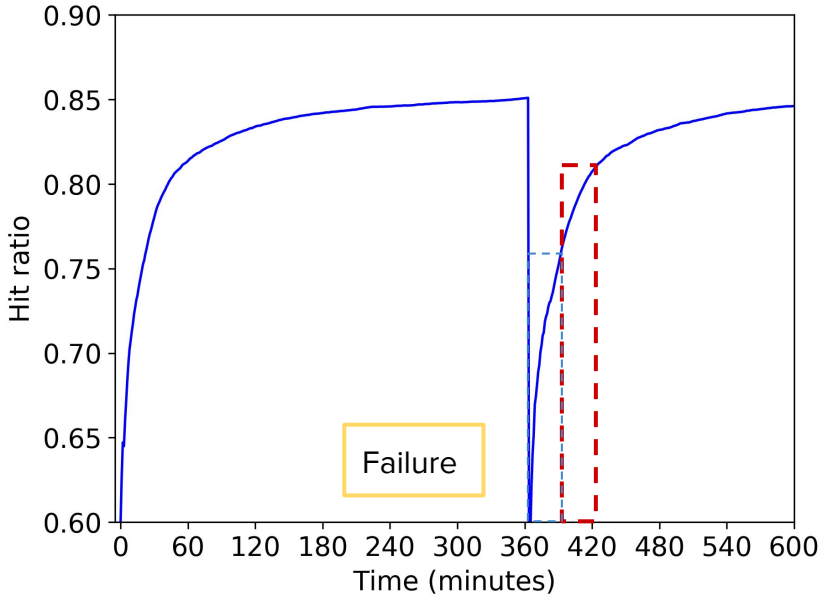




- low hit ratios for the first 30 minutes due to cold misses

Scenarios for a cold cache





- low hit ratios for the first 30 minutes due to cold misses
- improved hit ratios in the next 30 minutes as cache warms up

Problem Description

Given a hypervisor-managed storage I/O cache,

- how to reduce the time it takes to warm up the cache?
  - how to get to high hit ratio quickly?
- how to measure the effectiveness of prewarming?
  - metrics for cost & benefit?

Base model: Nutanix DSF

- 3 hashmaps maintained by the distributed filesystem

Hashmap 1

Maps the SHA1 hash of vDisk ID and vBlock no. to an egroup (extent group) ID.

➔ One entry for each 1 MB data (vBlock) of a vdisk
➔ Maps to an egroup file which stores the actual data

Hashmap 3

Maps an egroup ID to a list of slices (32 KB chunks) and their offset into the egroup file

➔ One entry for each 4 MB data stored across the fabric
➔ vBlocks from different vdisks make an egroup

vBlock = extent

Base model: Nutanix DSF

- we make two queries to the metadata store:
  - HM1 object query to get egroup ID
  - HM3 object query to get to actual disk block

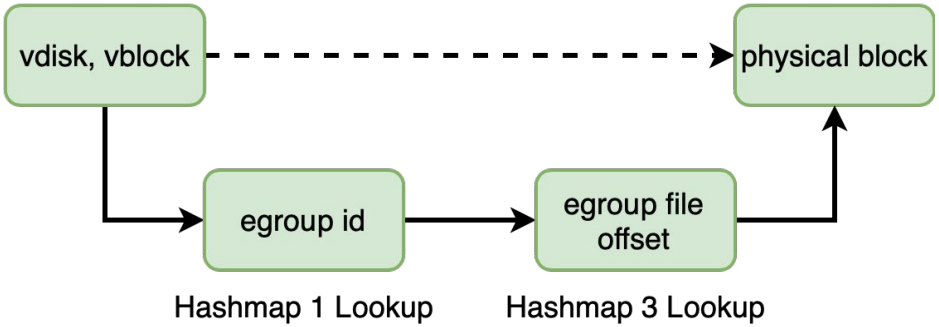


Fig: Metadata translation in the Nutanix DSF

Outline

I. **Cache Mechanics**

II. Workload Characterization

III. Empirical Analysis

Cache mechanics: persisting

- involves saving the cache state at a particular instant to the disk
  - we dump the state after every *n* requests served by the cache
  - doing this periodically creates a series of snapshots

Having a set of *m* snapshots, need to analyze them to determine the important objects.

We take a snapshot for each vDisk and one for all HM3 objects.

Each snapshot is represented as a bitmap saved as a binary file to the disk

Cache mechanics: persisting

```
HM1 (VDISK 0)          HM1 (VDISK 1)          HM3
-------------------    -------------------    ----------------------
1  0  0  0  0  0  1    0  1  0  0  0  1  0    1  0  1  1  0  1  1  0  Snapshot 1
0  0  0  1  0  0  1    0  0  0  1  1  0  1    0  0  1  1  1  0  1  1  Snapshot 2
1  1  0  0  1  0  1    1  1  0  0  0  0  1    0  1  0  0  0  0  1  0  Snapshot 3
.  .  .  .  .  .  .    .  .  .  .  .  .  .    .  .  .  .  .  .  .  .     ...
1  0  1  0  0  1  0    0  0  1  0  0  1  0    0  0  1  0  1  0  0  1  Snapshot m
-------------------    -------------------    ----------------------
1  0  1  1  1  0  0    0  0  0  0  1  0  1    0  0  1  0  1  1  1  1  Prewarm set
-------------------    -------------------    ----------------------
```

- each bit represents a metadata object (only key part, not value)
  - to get the actual object, need to query metadata store
- on bit (1) means object was cached at time of taking snapshot

Cache mechanics: persisting

- Snapshot rate: how frequently do we save cache state to disk?
  - a snapshot can be taken after every *x* requests, or in every *n* seconds

Pros:
* Low overhead as less writes happen to the disk
* Less disk space used
* Faster analysis as there are less files to analyze

Cons:
* Inability to capture fine-grained changes in cache state

Pros:
* Changes in cache state are captured more frequently enabling a finer analysis

Cons:
* Huge amount of disk space used
* Overhead due to saving cache state and loading for analysis as there are a lot of snapshot files

Cache mechanics: analyzing snapshots

- need to look at a history of snapshots to determine important objects
  - sorted by importance, these objects will be used for prewarming
    - this constitutes the prewarm set
  - various heuristics available to define the “importance”
    - frequency, recency, or both?
- we maintain per-object score to define their importance
  - prewarm set will contain objects in descending order of score
  - scores for prewarm set are global
    - they are considered across all HM1 & HM3 objects

Cache mechanics: analyzing snapshots

k-Frequent

```
1  0  0  0  0  1  1  0  0  1   Snapshot 1
0  0  0  1  1  0  1  1  0  0   Snapshot 2
1  1  0  0  0  0  1  0  1  1   Snapshot 3
0  0  0  1  1  1  0  1  1  1   Snapshot 4
1  0  1  0  1  1  0  1  1  0   Snapshot 5
--------------------------
3  1  1  2  3  3  3  3  3  3   Frequency score
--------------------------


--------------------------
3  3  3  3  3  3  3  2  1  1   Frequency values
0  4  5  6  7  8  9  3  1  2   Objects sorted by frequency
--------------------------
1  1  1  1  1  0  0  0  0  0   Memory constraint
--------------------------     (~size of 5 objects)

--------------------------
1  0  0  0  1  1  1  1  0  0   Prewarm set
--------------------------
```

k-Recent

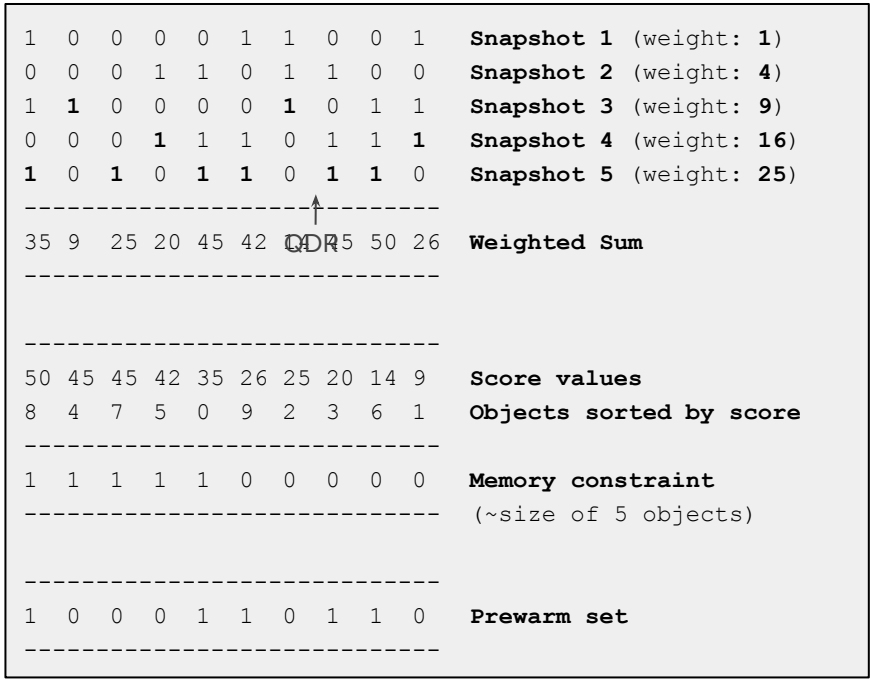
```
1  0  0  0  0  1  1  0  0  1   Snapshot 1
0  0  0  1  1  0  1  1  0  0   Snapshot 2
1  1  0  0  0  0  1  0  1  1   Snapshot 3
0  0  0  1  1  1  0  1  1  1   Snapshot 4
1  0  1  0  1  1  0  1  1  0   Snapshot 5
--------------------------
5  3  5  4  5  5  3  5  5  4   Recency score
--------------------------


--------------------------
5  5  5  5  5  5  4  4  3  3   Recency values
0  2  4  5  7  8  3  9  1  6   Objects sorted by recency
--------------------------
1  1  1  1  1  0  0  0  0  0   Memory constraint
--------------------------     (~size of 5 objects)

--------------------------
1  0  1  0  1  1  0  1  0  0   Prewarm set
--------------------------
```

Cache mechanics: analyzing snapshots

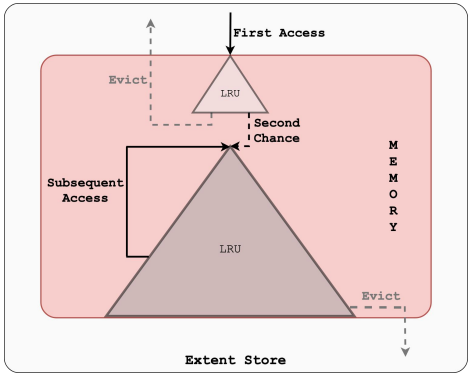
k-Frerecent



LNR = Linear, QDR = Quadratic

Cache mechanics: prewarming

- involves iterating over prewarm set and loading the objects proactively into cache
  - load most important objects first, into multi-touch pool
    - if the pool gets full, move on to single-touch
  - skip objects that are already cached
  - stop prewarming if cache gets full or no objects remain in the set
- need to know how many of these objects to load
  - and, how fast to load these objects



Cache mechanics: prewarming

- we define two parameters essential to prewarming process
  - prewarm size limit & prewarm rate
- Prewarm size limit: max. space available in cache to perform prewarming
  - we specify this limit as %age of the total cache size
- Prewarm rate: max. size objects to load in a second of time
  - we specify this rate in MB/s

Outline

I. Cache Mechanics

II. Workload Characterization

III. Empirical Analysis

Workloads for storage I/O

- need vDisk workloads as input to the cache
  - they are a series of (pre-recorded) block I/O requests
  - each of these requests has
    - vDisk ID
    - sector number
    - I/O size
    - timestamp
  - type of request (read/write) not important as we only cache metadata reads

Workloads for storage I/O

- two type of workloads:
  - Synthetic & Real

- Synthetic: generated using a tool/benchmark
  - may not correspond to an actual vDisk traffic
  - may not originate from even an actual filesystem
- Real: recorded on a system used for normal operations
  - I/O tracing mechanisms used on disk partitions/mount points

Workloads: real

- used block layer I/O tracing on a set of vDisks
  - common workloads such as mail, database, web server etc.
  - VMs running on IITB CSE Department infrastructure
  - negligible overhead due to tracing
  - non-intrusive method; no information about page cache available
- tracing was done using the *blktrace* tool
  - takes a disk partition as input
  - captures block I/O requests from the kernel

Workloads: real

- we capture only the *Issued* requests
- requests are for sector numbers
  - offset = sector * 512 byte
- we parse the requests using the *blkparse* tool
- most requests originate from kworker threads and journaling daemon

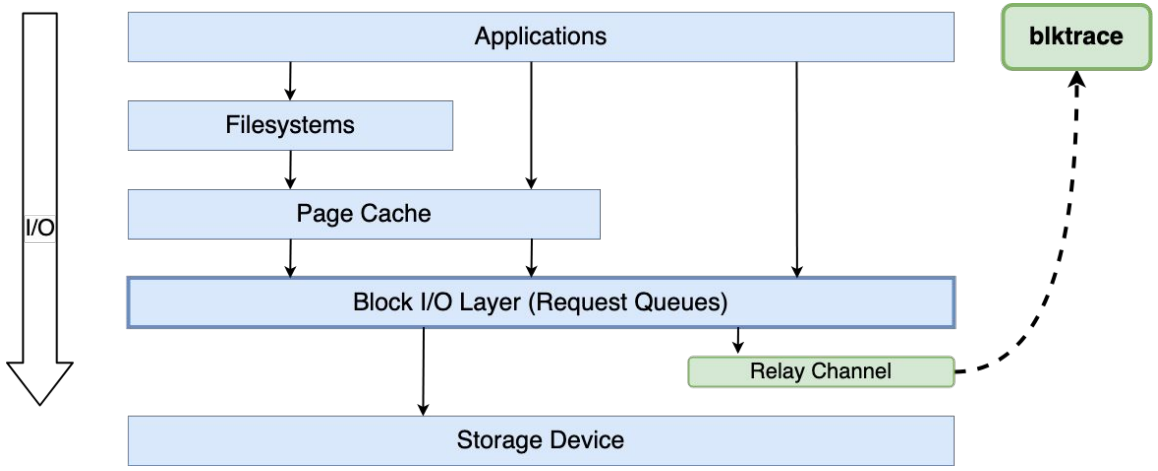
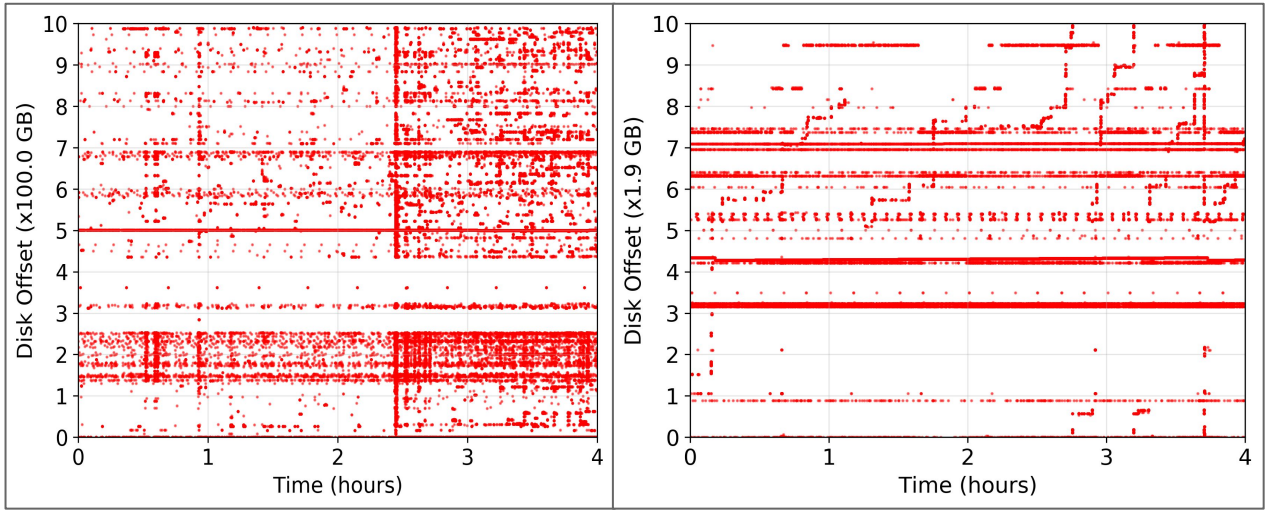


Fig: Block Layer I/O Tracing using blktrace

Workloads: real

- we label the real workloads collected using this method as RealCSE

vDisk	Size (GB)	Avg. IOPS
mail1	100	13.93
mail2	500	19.99
mail3	1000	8.89
db	19	2.54
ldap	19	2.12
web1	19	2.19
web2	19	1.08
web3	16	0.45



mail3 web1

Fig: Access pattern of a workload generated from two of the vDisks

Workload: synthetic

- created a tool for generating such workloads
  - it does not record I/O requests on an actual vDisk filesystem
  - generates a series of I/O requests based on certain parameter configurations
    - average IOPS for each second [Normal(μ, σ)]
    - sector numbers for each request [Uniform(min, max)]
    - I/O size for each request [weighted uniform distribution]
  - makes use of “hotspots” to divide disk space into regions of high activity
  - created multiple such configurations to vary the block (sector) access pattern

Workload: synthetic

Example of one such configuration:

- creates I/O traffic with avg. 100 requests per second
- divides the disk into 5 hotspots
  - we specify [% disk space, % I/O] to distribute the requests
- cluster is a group of sectors accessed sequentially (*locality*)
  - cluster stride indicates the distance between consecutive sector accesses (in units of sectors)
    - an avg. distance of 500 sectors here
  - cluster size indicates how many such consecutive sectors to access
    - an avg. of 5 sectors in a cluster

```
iops:
      randvar: normal
      mean: 100
      stddev: 20
sector:
      size: 512b
      cluster:
            size:
                  min: 1
                  max: 10
            stride:
                  min: 1
                  max: 1000
hotspots:
      - [10%, 1%]
      - [10%, 1%]
      - [30%, 10%]
      - [40%, 20%]
      - [10%, 1%]
iosizes:
      - [80%, 4096b]
      - [10%, 8192b]
      - [5%,  16384b]
      - [5%,  131072b]
```

Workload: synthetic

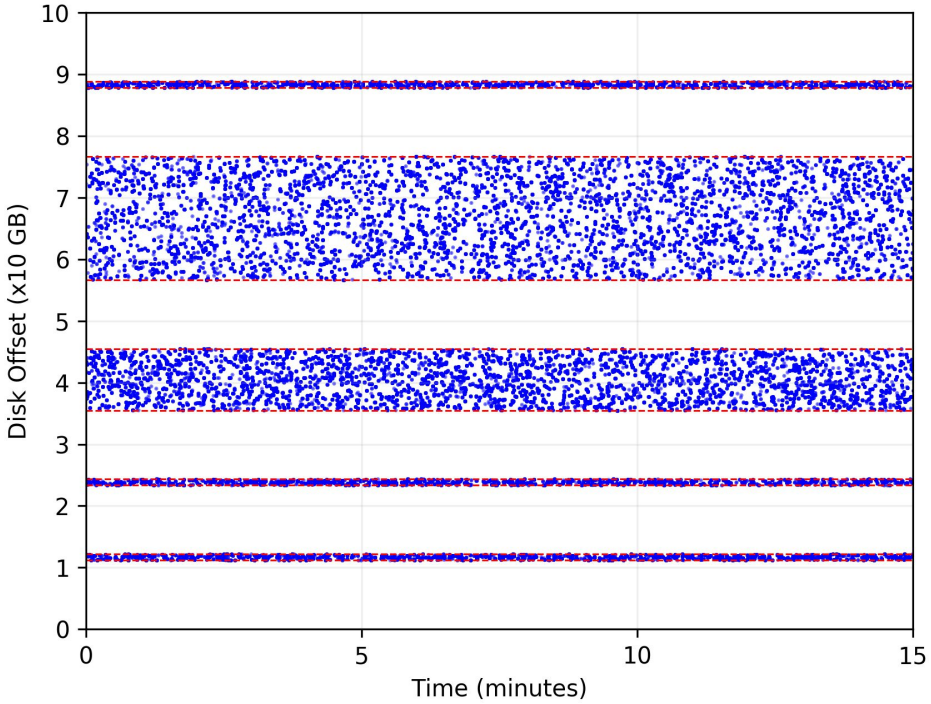


Fig: Access pattern of a workload generated using the example configuration

```
iops:
      randvar: normal
      mean: 100
      stddev: 20
sector:
      size: 512b
      cluster:
            size:
                  min: 1
                  max: 10
            stride:
                  min: 1
                  max: 1000
hotspots:
      - [10%, 1%]
      - [10%, 1%]
      - [30%, 10%]
      - [40%, 20%]
      - [10%, 1%]
iosizes:
      - [80%, 4096b]
      - [10%, 8192b]
      - [5%,  16384b]
      - [5%,  131072b]
```

Workload: synthetic

- generated two workload sets using the custom tool
  - we label them as Syn6GB & Syn4GB
    - ‘6’ and ‘4’ here denote the total cache size we used for these workloads in experiments
- all vDisks in these sets:
  - have same size (100 GB)
  - their avg. IOPS range from 25 to 200
  - have different disk block access patterns

Workloads: characterization

We have categorized these workloads on various parameters in the table below

Workload Set	Type	Avg. IOPS	IOPS distribution	Spatial Locality	Temporal Locality	wss80
Syn6GB	Synthetic	Medium	Uniform	Medium	Low	High
Syn4GB	Synthetic	Medium	Uniform	Medium	Low	Medium
RealCSE	Real	Low	Bursty	High	High	Low

- information about IOPS is known for synthetic workload beforehand
  - for the RealCSE workload, we observed their access pattern
- we ran a few experiments for determining wss80
  - this wss80 is different from the runtime wss in memory
  - it is the minimum cache size which guarantees us a hit ratio of 0.8

Outline

I. Cache Mechanics

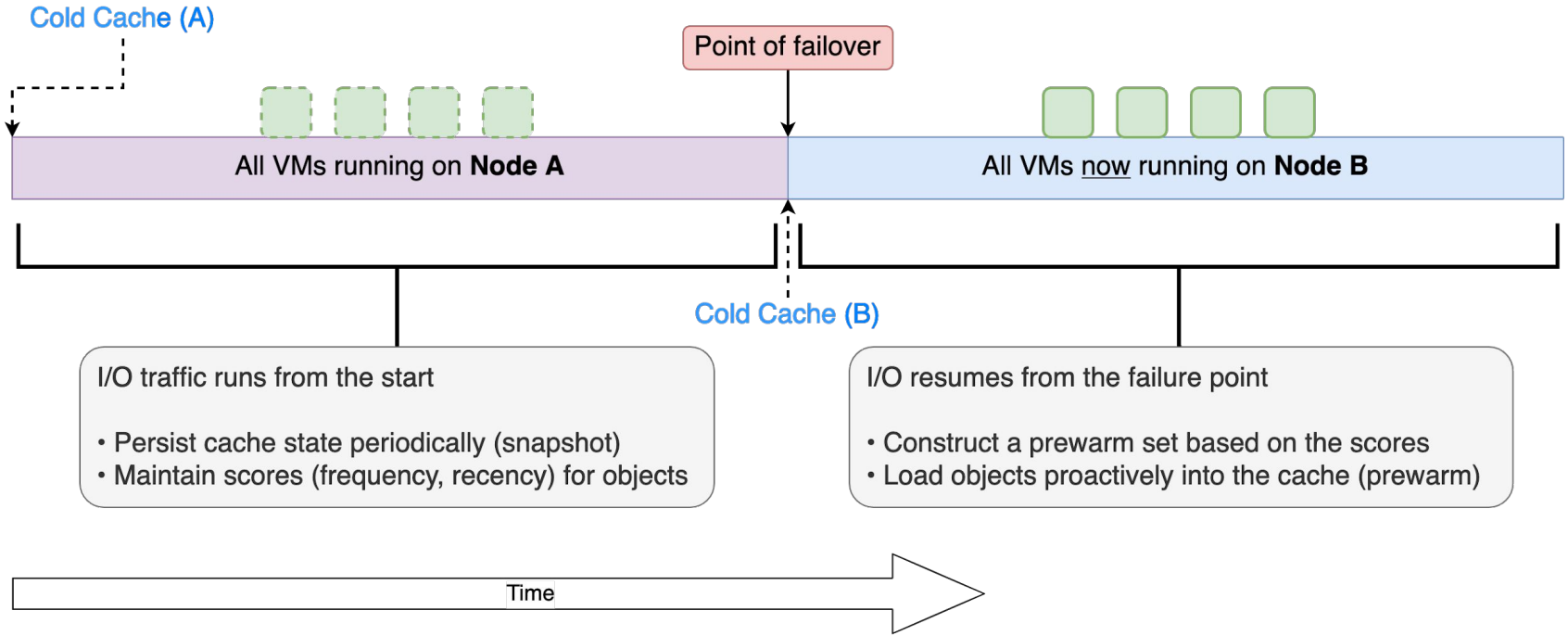
II. Workload Characterization

III. Empirical Analysis

Experiment Scenario

- multiple VMs are running on a (source) node which experiences failure
  - all VMs have only one vDisk
- these VMs are then migrated to the same (dest.) node which is idle
  - the dest. cache is cold whereas the cache on source was warm
- prewarming is started for each vDisk before its I/O traffic starts
- *we assume here that the failover migration is instantaneous*

Experiment Scenario



Experiment: parameters

- we have identified some parameters which are relevant to our prewarming process
- throughout the experiments, we vary only some of these while keeping the other fixed
- parameters which we have varied are more interesting than the others at this stage of study

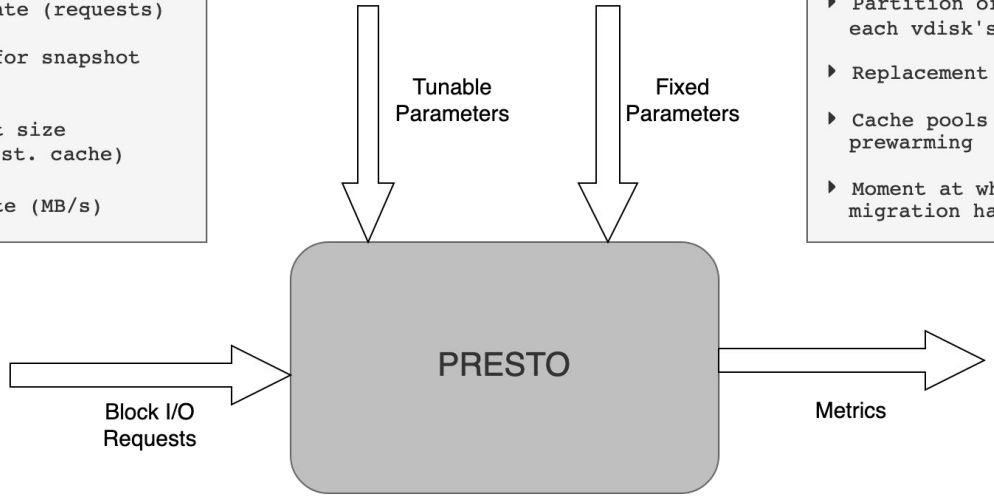
Experiment: parameters

Tunable Parameters

- Snapshot rate (requests)
- Heuristic for snapshot analysis
- Prewarm set size (%age of dest. cache)
- Prewarm rate (MB/s)

Fixed Parameters

- Total cache size
- HM1 & HM3 object sizes
- Number of VMs/vDisks
- Total duration of I/O requests
- Partition of prewarm set for total HM1 and HM3 objects
- Partition of prewarm set for each vdisk's HM1 objects
- Replacement Policy
- Cache pools considered for prewarming
- Moment at which failover/ migration happens



Experiment: parameters

- tunable parameters:
  - snapshot rate and heuristic used affect the source node
    - they eventually have a huge impact on the dest. node as well
  - prewarm size limit and rate affect the dest. node

Experiment: parameters

We consider the following set of values for these parameters:

- snapshot rate: 10k, 50k, 100k (requests)
- heuristic: k-Recent, k-Frequent, k-Frerecent
- prewarm set size limit: 5%, 10%, 25%, 50%, 75%, 100%
  - %age of the cache on dest. node
- prewarm rate: 50 MBps, 100 MBps, 500 MBps, ∞

Experiment: for wss80 parameter

We first run a simple experiment to get an idea about the locality this vDisks in this workload exhibit.

- we change the cache size to observe the average hit ratio
- we run each vDisk individually on the cache, starting with 1 MB size
- if the vDisk was able to achieve 0.8 average hit ratio (over 4 hours), we stop the experiment for it

We decide the value for cache sizes for workloads from their wss80 we observed.

We also observed from experiment results that 1 MB worth of objects in cache represent 36 MB of the vDisk space

Experiment: metrics

- we make use of hit ratio as the metric for analyzing the prewarming effectiveness
- there are various costs associated with prewarming as well; most important ones are:
  - performance degradation (on source) due to snapshotting and analysis
  - latency (on dest.) due to cache miss penalty
  - increased network traffic (on dest.) due to metadata object fetch
- in the current study, we have not considered the cost metrics while determining the effectiveness

Experiment: metrics

- we consider 4 metrics based on the hit ratio (HR) on the dest. node after failure:
  - instantaneous HR: within 1 second
  - 5-min-avg HR: average of first 5 minutes
  - 15-min-avg HR: average of first 15 minutes
  - 30-min-avg HR: average of first 30 minutes

Experiment: key questions

- we try to answer the following questions for the 4 parameters:
  - how does changing the parameter value affect the hit ratios?
  - what is a reasonable value for the parameter which gives significant benefit?
- we perform an experiment for each of these parameters, for each workload

Experiment: Workload - Syn6GB

We first ran a basic experiment to calculate the wss80 values.

The collective wss80 value was ~7350 MB.

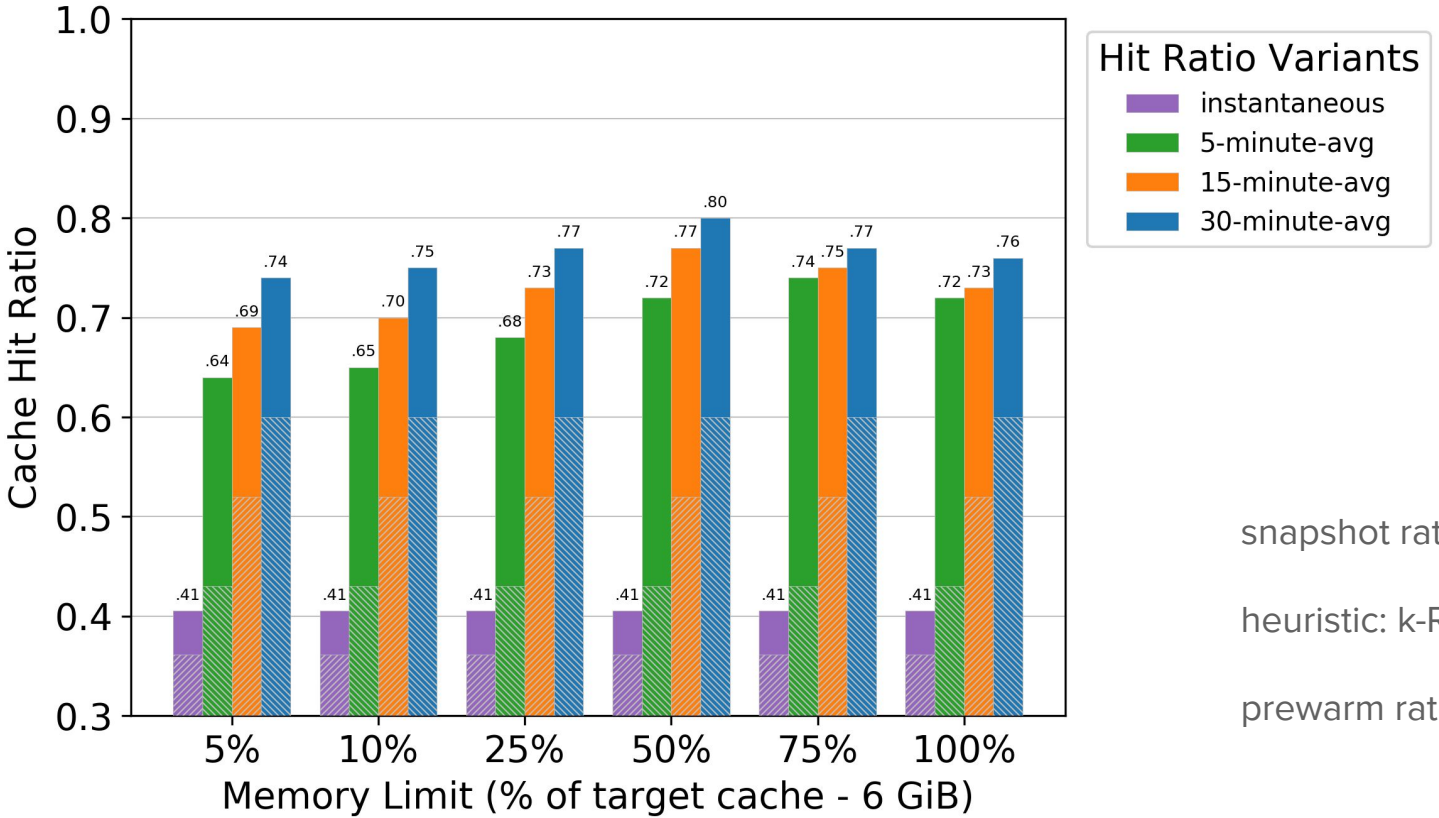
vDisk	Size (GB)	avg. IOPS (till failure)	wss80 (MB)	Data represented by wss80 (GB)	Runtime WSS (MB)
syn520	100	49.46	2000	70.31	684.53
syn521	100	24.61	425	14.94	123.03
syn539	100	100.03	250	8.79	257.94
syn540	100	149.27	2200	77.34	1612.74
syn554	100	199.4	175	6.15	317.93
syn570	100	149.92	2300	80.86	1318.43

Experiment: Workload - Syn6GB

Impact of prewarm set size limit on the cache

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	5%	100 MBps	k-Recent	0.41	0.64	0.69	0.74
100 K	10%	100 MBps	k-Recent	0.41	0.65	0.7	0.75
100 K	25%	100 MBps	k-Recent	0.41	0.68	0.73	0.77
100 K	50%	100 MBps	k-Recent	0.41	0.72	0.77	0.8
100 K	75%	100 MBps	k-Recent	0.41	0.74	0.75	0.77
100 K	100%	100 MBps	k-Recent	0.41	0.72	0.73	0.76

Experiment: Workload - Syn6GB



snapshot rate: 100K

heuristic: k-Recent

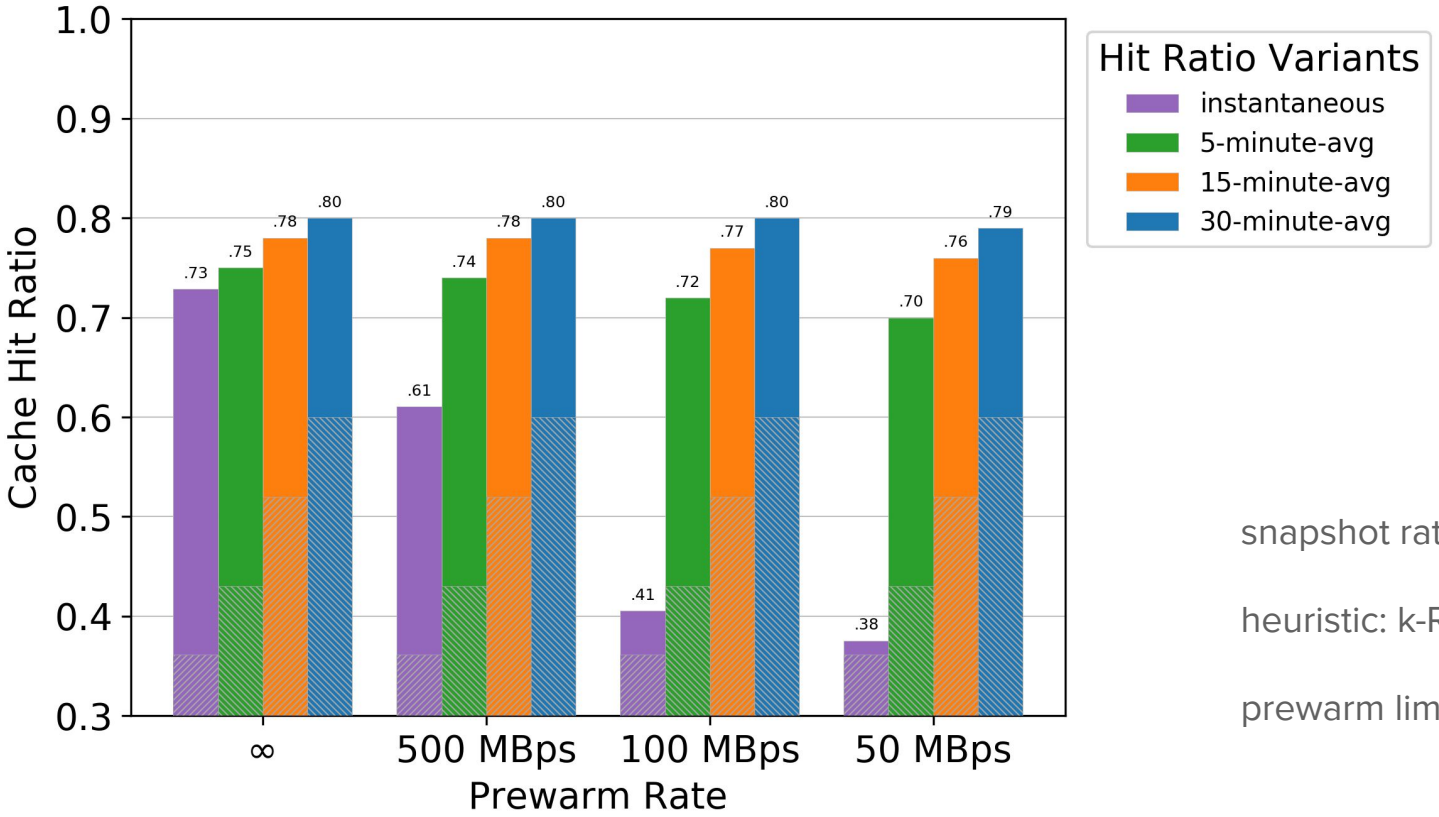
prewarm rate: 100 MBps

Experiment: Workload - Syn6GB

Impact of prewarm rate on the cache

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	50%	50 MBps	k-Recent	0.38	0.7	0.76	0.79
100 K	50%	100 MBps	k-Recent	0.41	0.72	0.77	0.8
100 K	50%	500 MBps	k-Recent	0.61	0.74	0.78	0.8
100 K	50%	inf	k-Recent	0.73	0.75	0.78	0.8

Experiment: Workload - Syn6GB



snapshot rate: 100K

heuristic: k-Recent

prewarm limit: 50%

Experiment: Workload - Syn6GB

Impact of snapshot rate on the cache

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
10 K	75%	100 MBps	k-Frequent	0.38	0.72	0.78	0.8
50 K	75%	100 MBps	k-Frequent	0.38	0.73	0.78	0.8
100 K	75%	100 MBps	k-Frequent	0.39	0.73	0.78	0.8

Experiment: Workload - Syn6GB

Impact of heuristic used for snapshot analysis on the cache

(with low prewarm size limit)

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	10%	50 MBps	k-Recent	0.38	0.64	0.7	0.75
100 K	10%	50 MBps	k-Frequent	0.38	0.51	0.59	0.65
100 K	10%	50 MBps	k-Frerecent (LNR)	0.38	0.51	0.58	0.65
100 K	10%	50 MBps	k-Frerecent (QDR)	0.38	0.51	0.58	0.65

Experiment: Workload - Syn6GB

Impact of heuristic used for snapshot analysis on the cache

(with high prewarm size limit)

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	75%	50 MBps	k-Recent	0.38	0.7	0.73	0.75
100 K	75%	50 MBps	k-Frequent	0.38	0.67	0.75	0.78
100 K	75%	50 MBps	k-Frerecent (LNR)	0.38	0.67	0.75	0.78
100 K	75%	50 MBps	k-Frerecent (QDR)	0.38	0.66	0.75	0.78

Experiment: Workload - Syn4GB

We ran the same set of experiments and observed in the results that the parameter values have the same effect relative to each other.

The hit ratio values vary in the same manner as in the experiments we did for Syn4GB workload.

Experiment: Workload - RealCSE

The collective wss80 value we observed here was ~105 MB

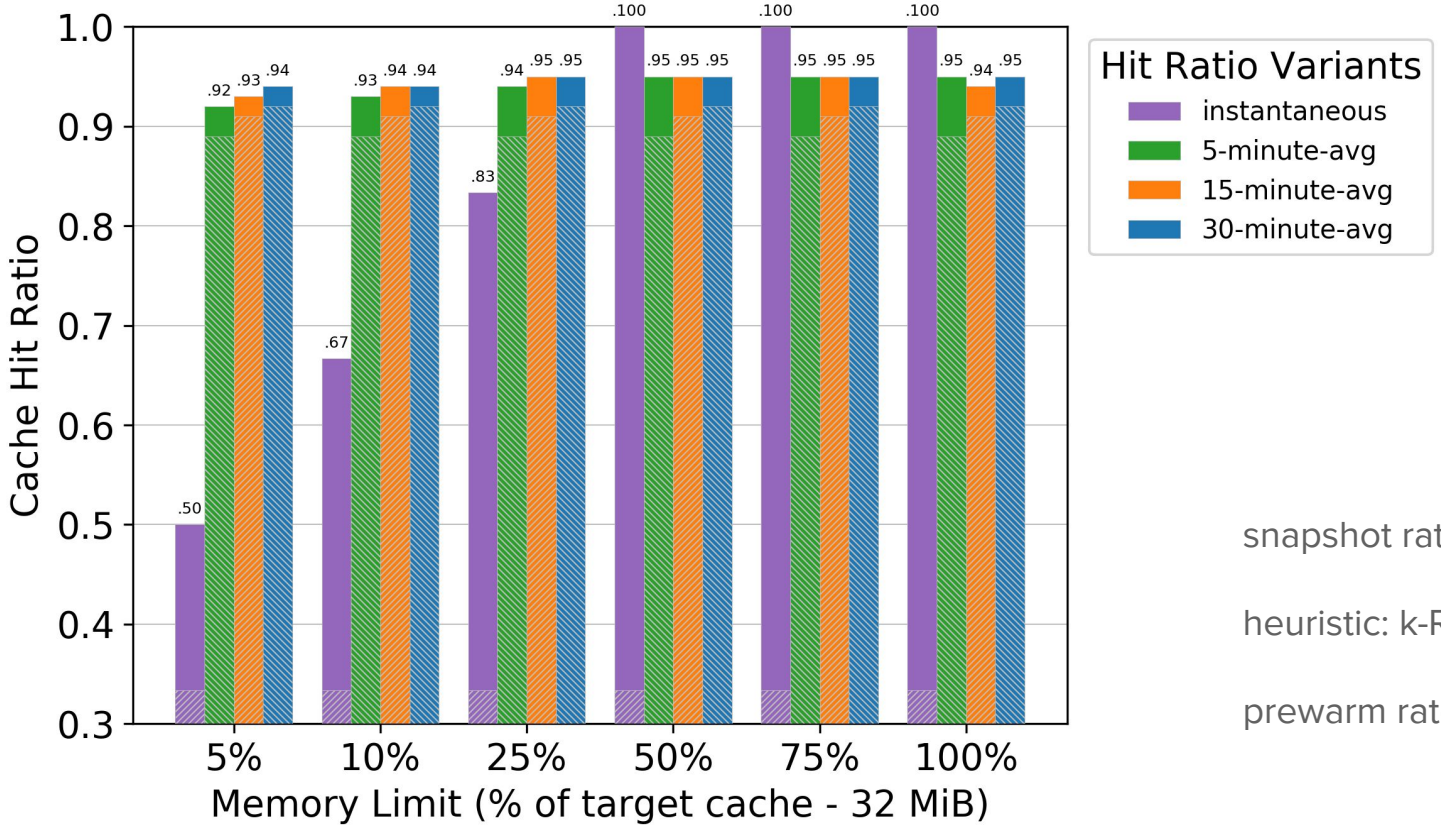
vDisk	Size (GB)	avg. IOPS (till failure)	wss80 (MB)	Data represented by wss80	Runtime WSS (MB)
mail1	5.18	100	5	0.18	4.76
mail2	15.15	500	1	0.04	6.43
mail3	3.03	1000	90	3.16	14.85
db	2.22	19	1	0.04	1.97
ldap	2.57	19	1	0.04	0.86
web1	1.81	19	5	0.18	1.63
web2	0.89	19	1	0.04	0.86
web3	0.39	16	1	0.04	0.64

Experiment: Workload - RealCSE

Impact of prewarm set size limit on the cache

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	5%	50 MBps	k-Recent	0.5	0.92	0.93	0.94
100 K	10%	50 MBps	k-Recent	0.67	0.93	0.94	0.94
100 K	25%	50 MBps	k-Recent	0.83	0.94	0.95	0.95
100 K	50%	50 MBps	k-Recent	1	0.95	0.95	0.95
100 K	75%	50 MBps	k-Recent	1	0.95	0.95	0.95
100 K	100%	50 MBps	k-Recent	1	0.95	0.94	0.95

Experiment: Workload - RealCSE



snapshot rate: 100K

heuristic: k-Recent

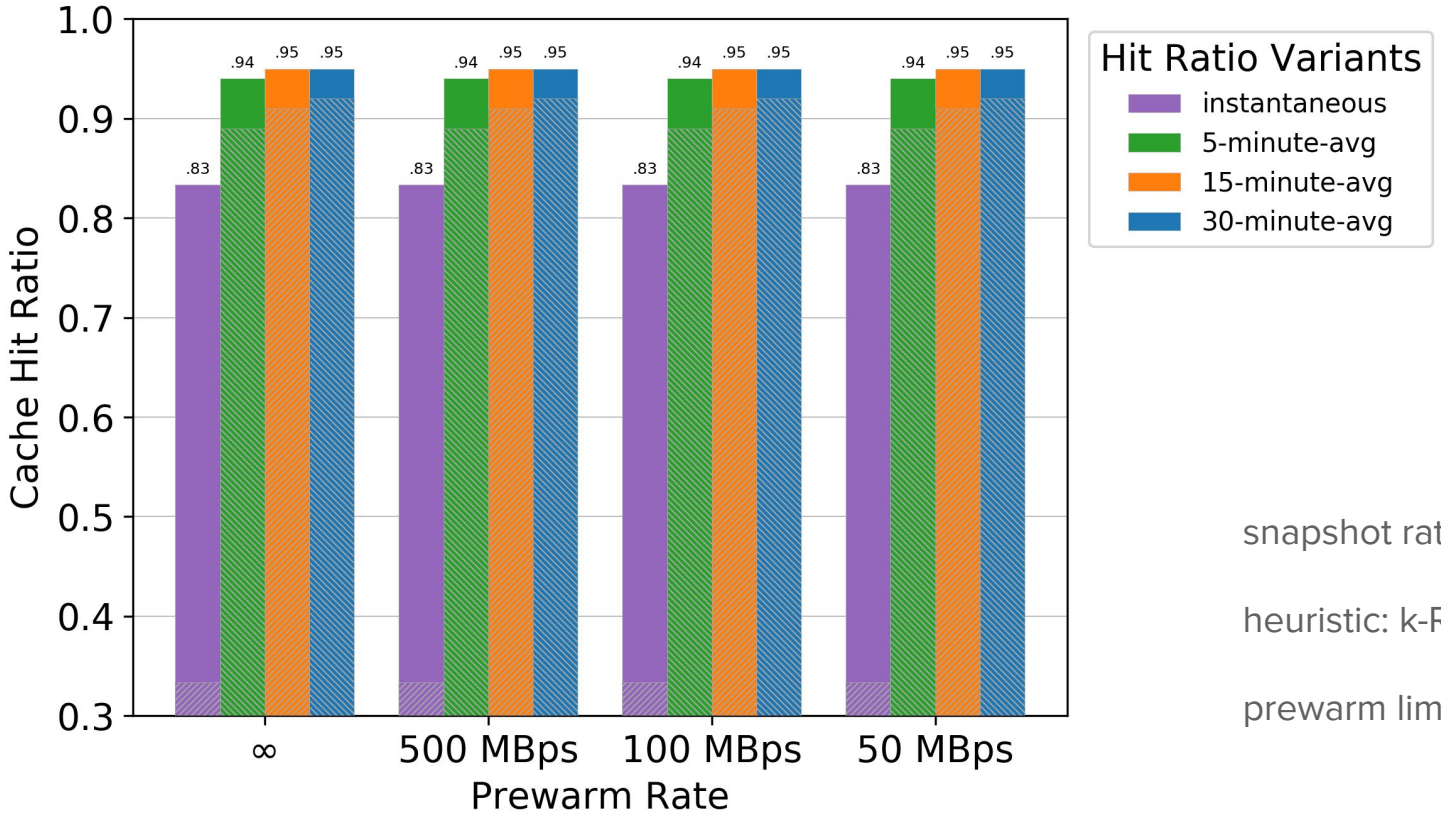
prewarm rate: 50 MBps

Experiment: Workload - RealCSE

Impact of prewarm rate on the cache

Snapshot Rate (requests)	Prewarm Set Size Limit (%age of 6GB)	Prewarm Rate	Heuristic	Inst. HR	5-min-avg HR	15-min-avg HR	30-min-avg HR
100 K	25%	50 MBps	k-Recent	0.83	0.94	0.95	0.95
100 K	25%	100 MBps	k-Recent	0.83	0.94	0.95	0.95
100 K	25%	500 MBps	k-Recent	0.83	0.94	0.95	0.95
100 K	25%	inf	k-Recent	0.83	0.94	0.95	0.95

Experiment: Workload - RealCSE



snapshot rate: 100K

heuristic: k-Recent

prewarm limit: 25%

Experiment: Workload - RealCSE

We observed from the results that changing the *snapshot rate* and *heuristic* for analysis had negligible impact on the hit ratio benefits on the dest. node.

Experiment: summary

- workload RealCSE had high locality of accesses
    - prewarming did not help much as the hit ratios were high even with a cold cache
- workload Syn6GB (and Syn4GB) showed a significant improvement in hit ratios, even with low prewarm size limits and low prewarm rates
- in most cases with Syn6GB, we saw that an increase in prewarm size limit and prewarm rate helped the hit ratios
- the choice of snapshot rate did not affect the results significantly, but the heuristic used did

Online Estimation of Parameters

- experiments performed by choosing parameters in advance
  - need to determine them in an online manner (while vDisk traffic is running)
- we know 3 important facts about a vDisk during the cache run:
  - Hit Ratio
  - avg. IOPS
  - WSS
- Is there a way to take this information and make decisions about various parameter values?

Online Estimation of Parameters

Some basic questions that come up at runtime:

- Should we prewarm a vDisk at all?
  - high locality -> no need
- How much should I prewarm the vDisk?
  - proportional to WSS
- At what rate to prewarm?
  - higher than the vDisk's I/O rate

Future Work

We need to add a mechanism by which prewarm set objects are given proper priority when evictions happen.

There are a few other directions left to explore:

- trying more diverse workload sets
- partitioning for the prewarm set
- number of vDisks running at the dest.
- latencies for miss penalty

Thank you!