# Improving IO latency of Containers by Adaptive Provisioning of Non Volatile Memory

**Shyamli Rao**
153050009

Under the guidance of
**Prof. Purushottam Kulkarni**

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

# Problem context

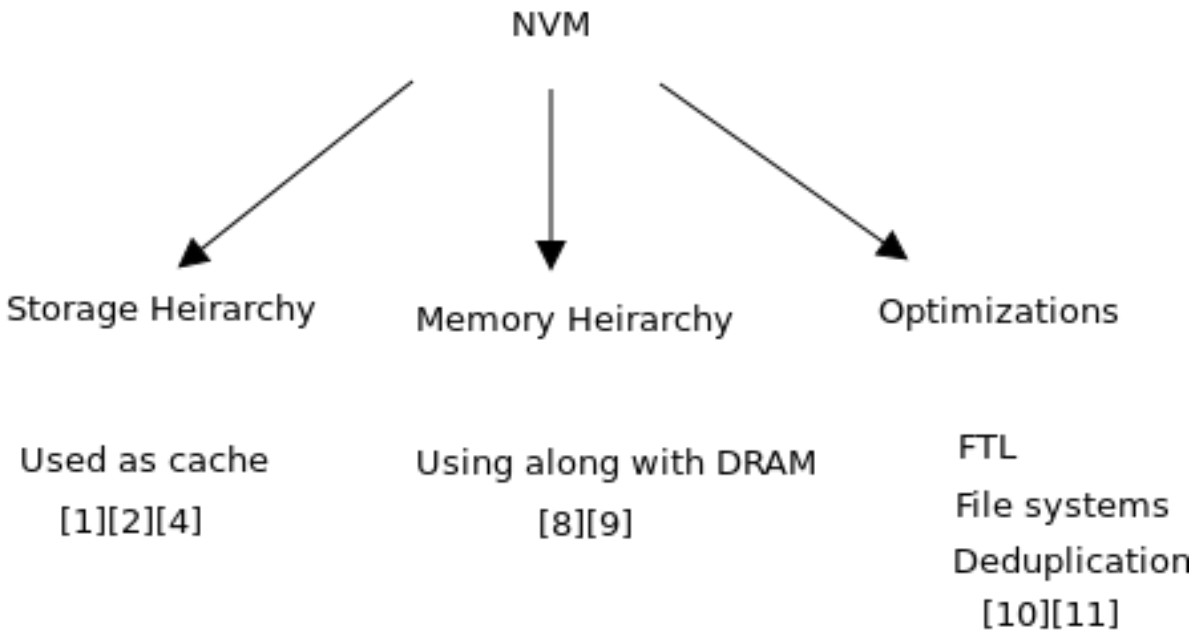
- Containers are emerging technology

- Nowadays NVMs are widely used

# Container

- Set of processes
- Isolated
- Resource control

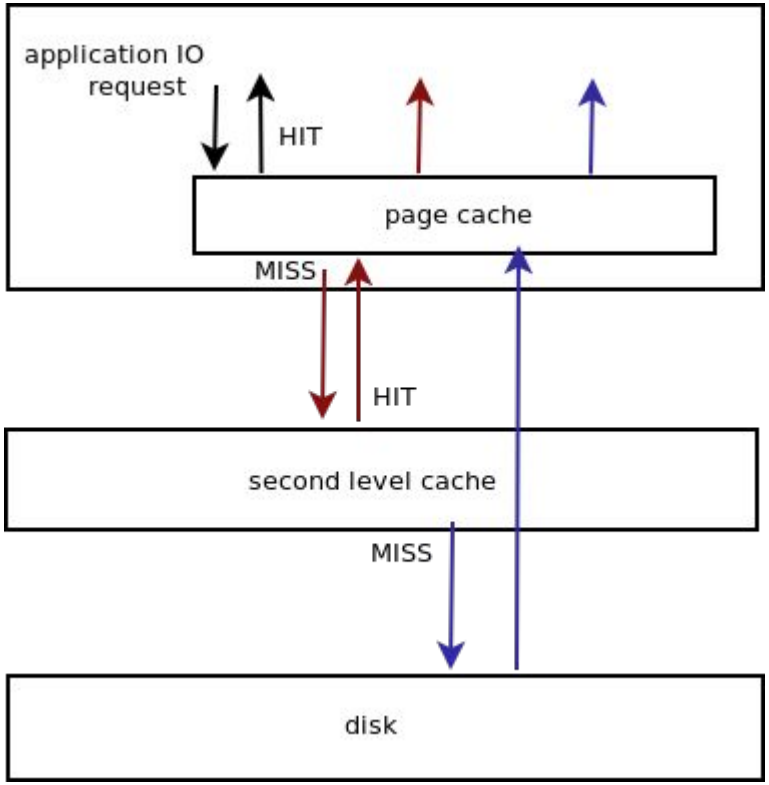
# NVM related work



# NVM as second level cache

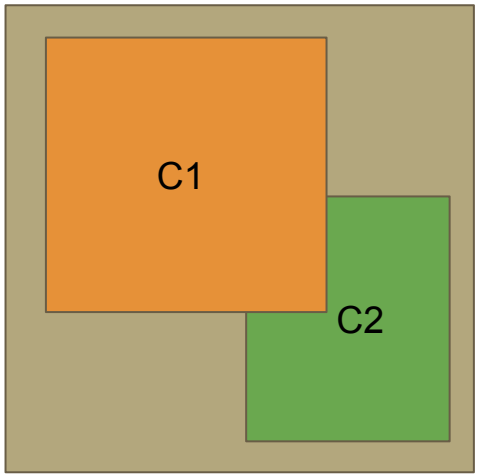
| | DRAM | Flash | HDD |
|---|---|---|---|
| Read(ns) | 10-50 | 25,000 | $5\text{-}8*10^6$ |
| Write(ns) | 10-50 | 200,000 | $5\text{-}8*10^6$ |
| Cost ($/GB)[1] | 8-12 | 1-3 | 0.05-0.10 |

Comparison of data storage technologies [1]



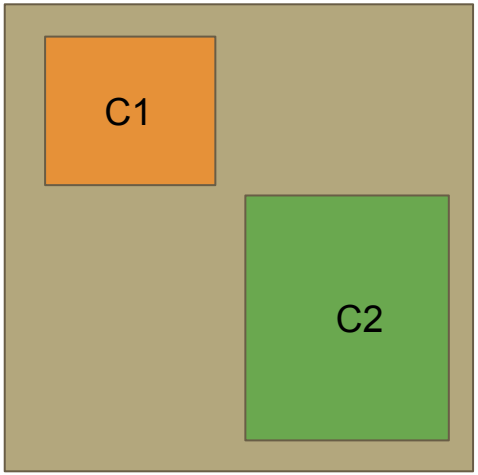
# Cache utilisation

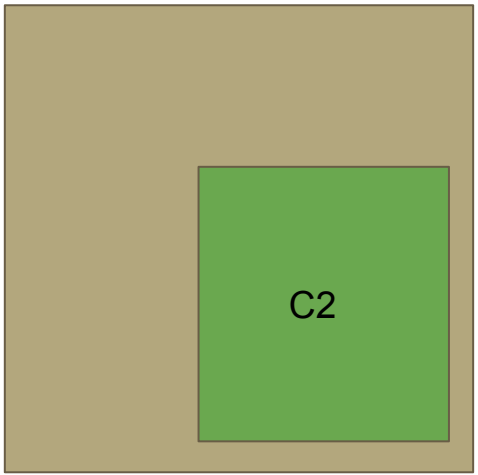
C1 : Container1
C2 : Container2




Current scenario

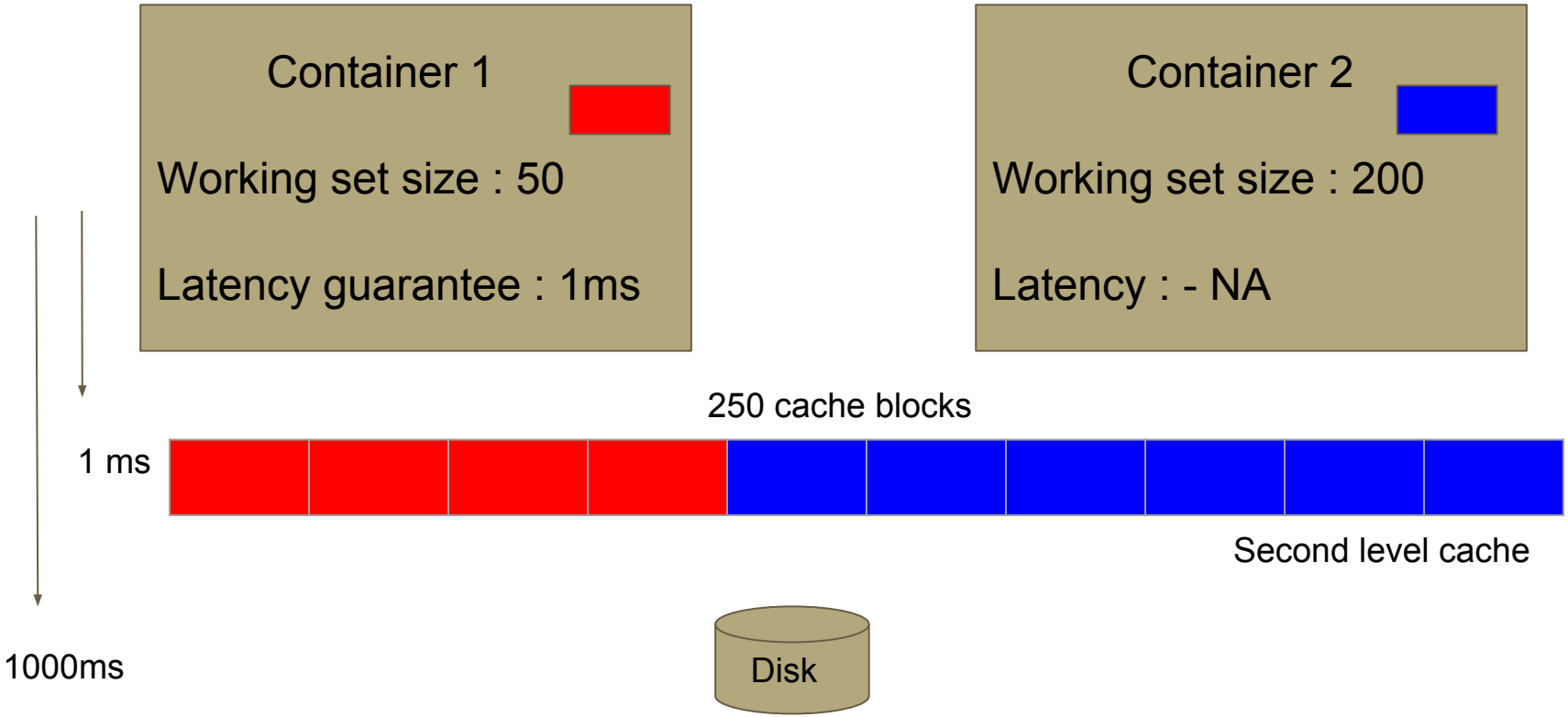
Cache occupancy




Desired scenario

# Satisfy SLA


Container 1
Working set size : 50
Latency guarantee : 1ms
Container 2
Working set size : 200
Latency : - NA
250 cache blocks
1 ms
Second level cache
1000ms
Disk

# Satisfy SLA

Container 1

Working set size : 50

Latency guarantee : 1ms

Container 2

Working set size : 200

Latency : - NA

250 cache blocks

# Satisfy SLA

Container 1

Working set size : 50

Latency guarantee : 1ms

Container 2

Working set size : 200

Latency : - NA

250 cache blocks

# Problem statement

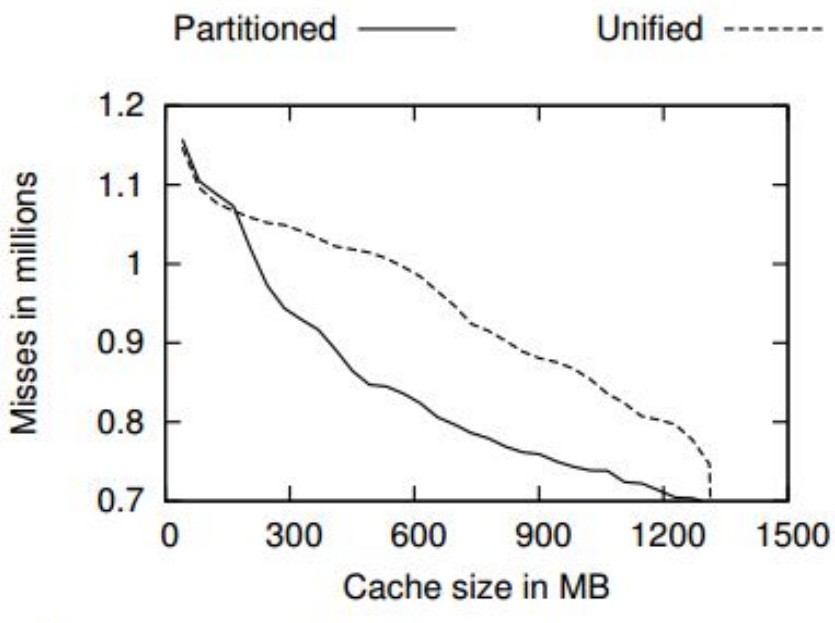
- Engineering a new design for
  - Partitioning SSD
  - Deciding policy

# Related work

# Existing solutions for VMs

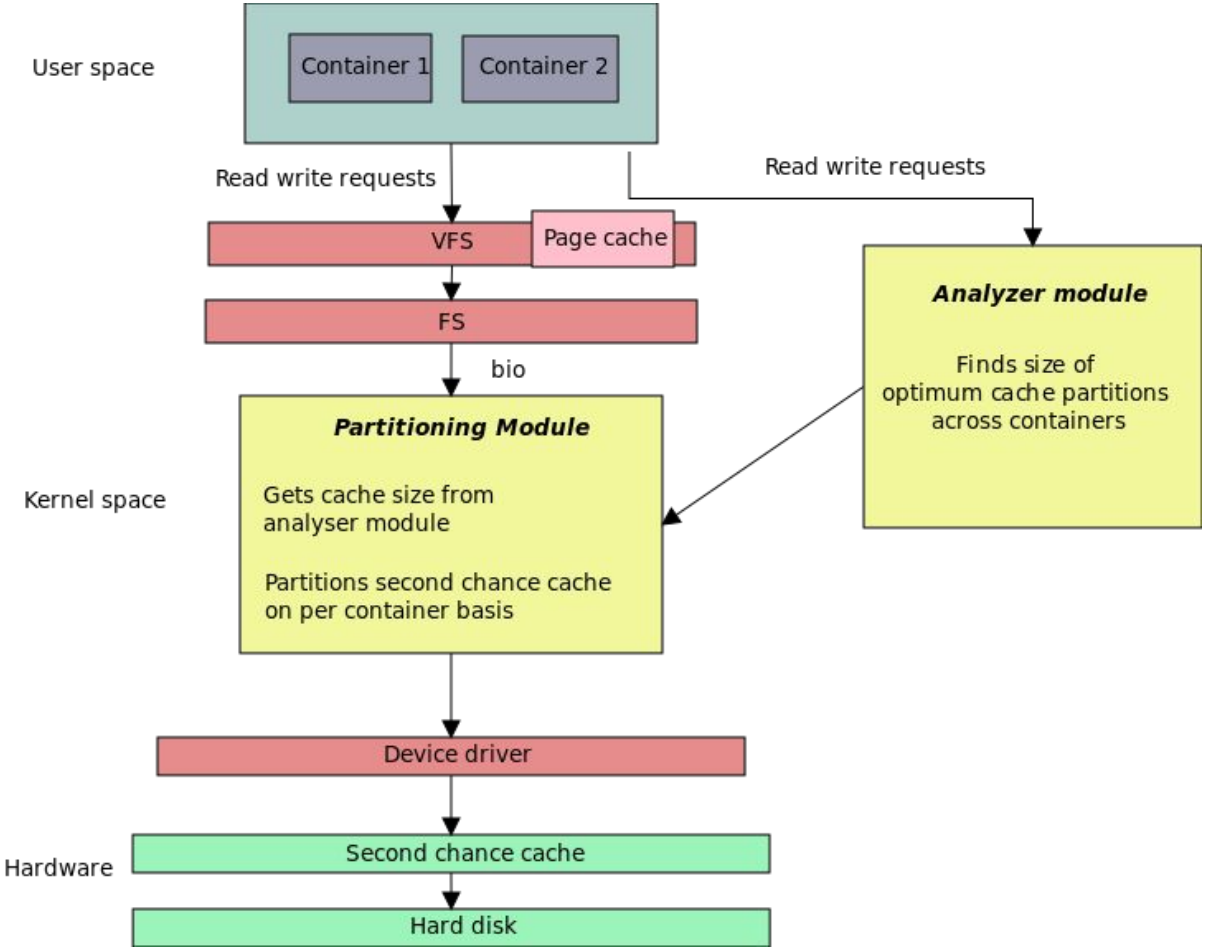
| Deciding policy | Fairness | SLA | Minimising overall latency | Partitioning |
|---|---|---|---|---|
| ExTmem[5] | NO | NO | YES | NO |
| vcacheShare[4] Multicache[1] | NO | NO | YES | Dynamic |
| 3-level[6] | YES | YES | NO | Dynamic |
| Unicache[7] | YES | NO | YES | Dynamic |
| Centaur[2] | YES | YES | YES | Dynamic |

# Unified cache vs Partitioned cache



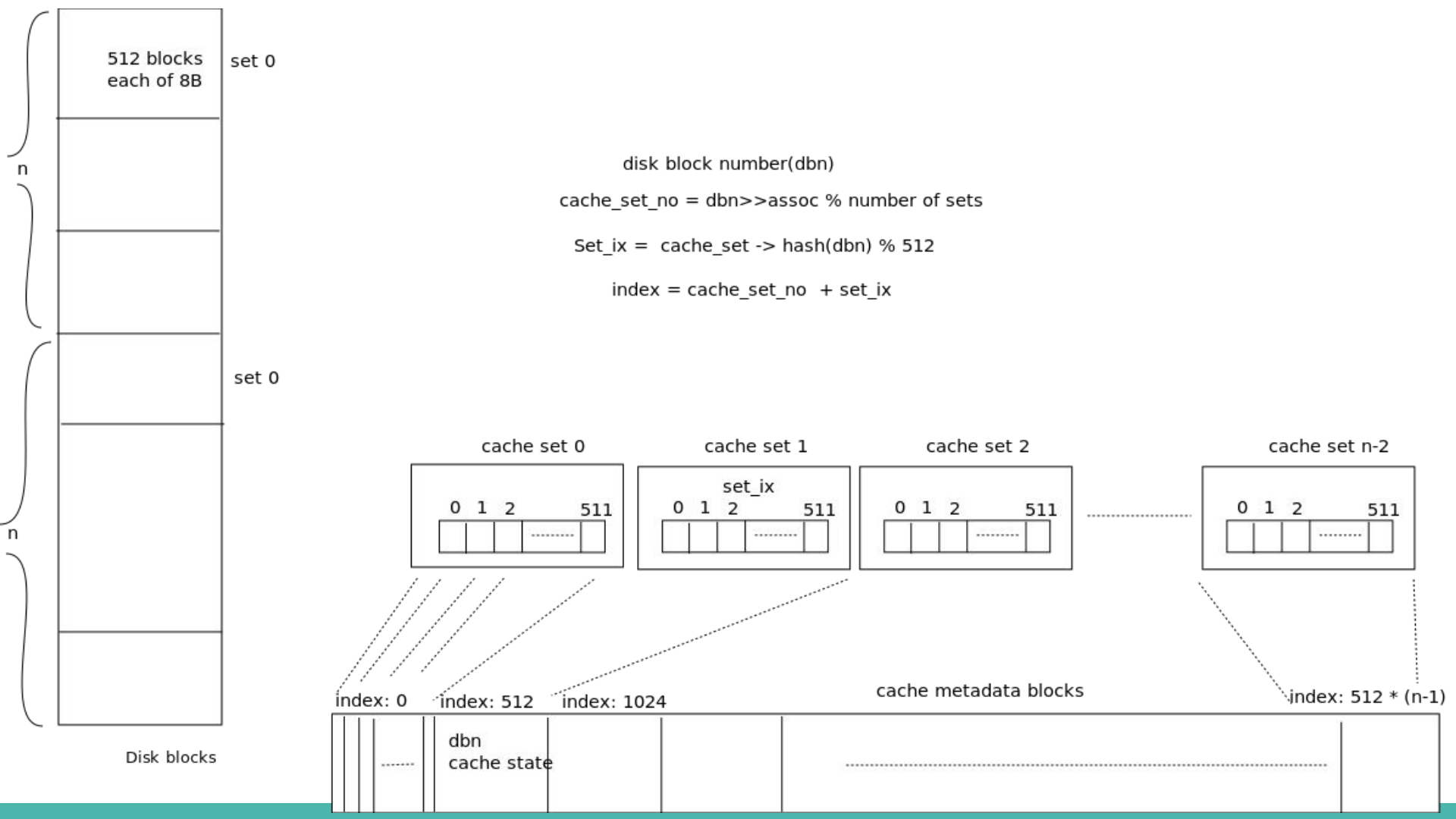

Unified vs Partitioned cache [2]

# Design


User space
Container 1
Container 2
Read write requests
Read write requests
VFS
Page cache
FS
bio
Analyzer module
Finds size of optimum cache partitions across containers
Kernel space
Partitioning Module
Gets cache size from analyser module
Partitions second chance cache on per container basis
Device driver
Hardware
Second chance cache
Hard disk

# Implementation

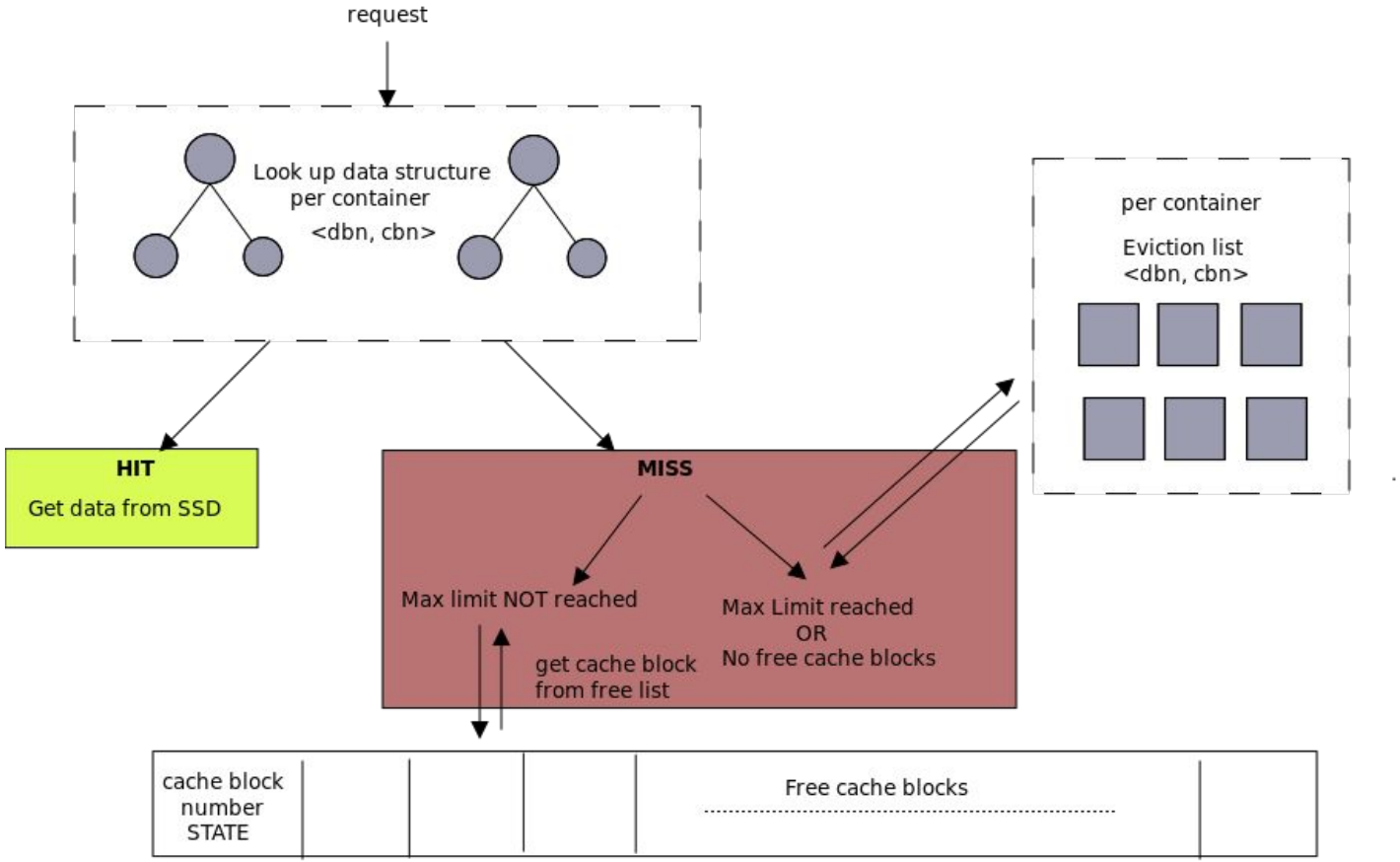
- Existing implementations for exposing SSD as cache :
  - dm-cache
  - Flashcache


512 blocks
each of 8B
set 0
n
set 0
n
Disk blocks
disk block number(dbn)
cache_set_no = dbn>>assoc % number of sets
Set_ix = cache_set -> hash(dbn) % 512
index = cache_set_no + set_ix
cache set 0
cache set 1
cache set 2
cache set n-2
set_ix
0 1 2 511
index: 0
index: 512
index: 1024
cache metadata blocks
index: 512 * (n-1)
dbn
cache state

# Partitioner

- Global list of unused cache blocks
- Per container
  - Lookup tree
  - Eviction list
  - Cache size limit

# IO Request handling



# Analyzer

Cache sizing can be based on

- Working Set size
- Priority
- SLA
- Fairness

# Current progress

- Provisioning Cache - Static partitioning
- Inclusive or Exclusive - Inclusive caching
- Write policy - Write bypass
- Cache eviction policy -FIFO

# Experiments

# Correctness verification

| | File size | No. of Blocks | 1st run | 2nd run with page cache | 3rd run without page cache |
|---|---|---|---|---|---|
| Experiment 1 | 8KB | 2 | MISS : 2<br>HIT : 0 | MISS: 0<br>HIT : 0 | MISS: 0<br>HIT : 2 |
| Experiment 2 | 1MB | 256 | MISS : 256<br>HIT : 0 | MISS: 0<br>HIT : 0 | MISS: 0<br>HIT : 256 |
| Experiment 3 | 8MB | 2048 | MISS : 2048<br>HIT : 0 | MISS: 0<br>HIT : 0 | MISS: 0<br>HIT : 2048 |

Inference : All the blocks are cached in SSD.

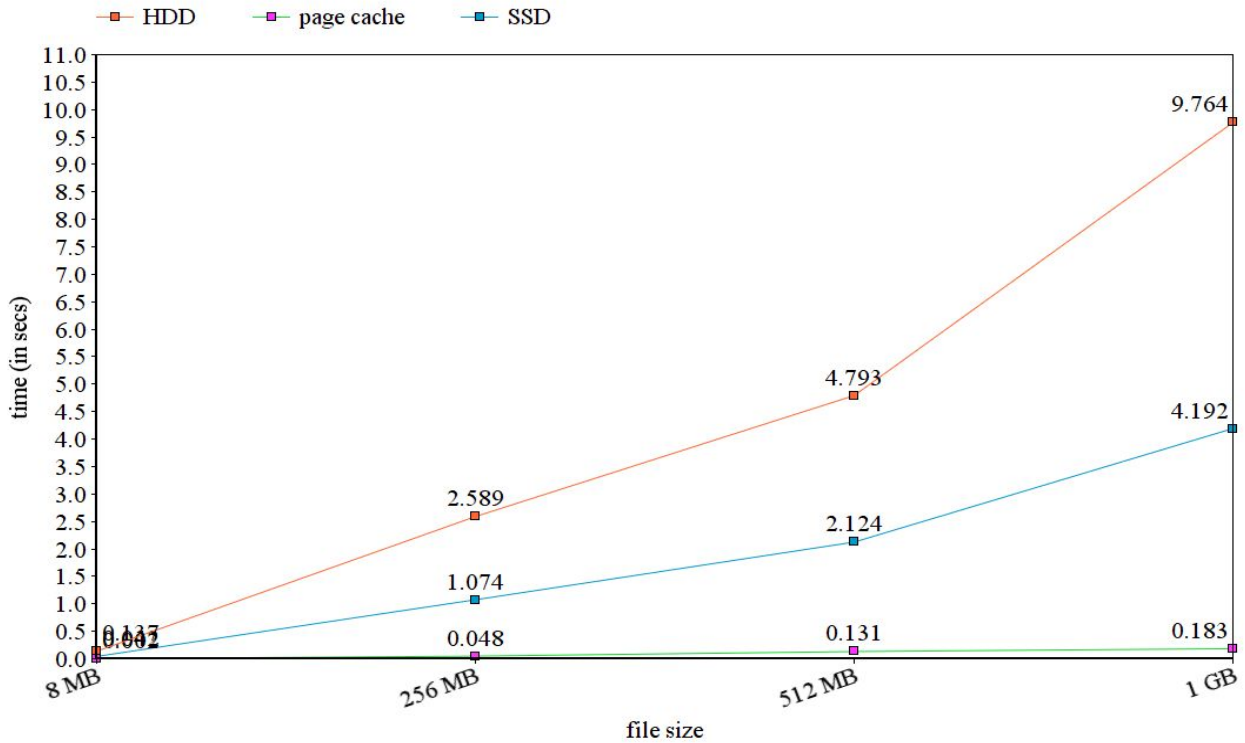
# Adhere to cache limit

Cache limit set to 1000

| | File size | No. of Blocks | 1st run | 2nd run with page cache | 3rd run without page cache |
|---|---|---|---|---|---|
| Experiment 1 Sequential | 8MB | 2048 | MISS : 2048<br>HIT : 0 | MISS: 0<br>HIT : 0 | MISS: 2048<br>HIT : 0 |
| Experiment 2 In loop of 60 blocks | 8MB | 2048 | MISS : 60<br>HIT : 0 | MISS: 0<br>HIT : 0 | MISS: 0<br>HIT : 60 |

Inference : Cache limit is adhered since as number of blocks exceed cache size they are replaced using FIFO replacement policy.

# IO latency




Inference : SSD reduces the latency by 49% with respect to disk.

# Conclusion & Future work

- Extend current implementation to partition dynamically
- Implement analyser
- Eviction policy

# References


1. *Sundaresan Rajasekaran, Shaohua Duan, Wei Zhang, Timothy Wood*. Multi-Cache: Dynamic, Efficient Partitioning for Multi-Tier Caches in Consolidated VM Environments.
2. *Ricardo Koller, Ali Jos´e Mashtizadeh, Raju Rangaswami*. Centaur: Host-side SSD Caching for Storage Performance Control. Proceedings of Autonomic Computing, 2015 IEEE International Conference on cloud computing, pages 966–973, 2015.
3. *Mohan Srinivasan*, Flashcache, Facebook
4. *Fei Meng , Li Zhou, Xiaosong Ma, Sandeep Uttamchandani and Deng Liu*. vCacheShare: Automated Server Flash Cache Space Management in a Virtualization Environment. Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference, pages 133–144, 2014.
5. *Vimalraj Venkatesan, Wei Qingsong, and Y. C. Tay*. ExTmem: Extending Transcendent Memory with Nonvolatile Memory for Virtual Machines. Proceedings of the 2014 IEEE Intl Conf on High Performance Computing and Communications, pages 51–60, 2014.

6. *Vimalraj Venkatesan, Wei Qingsong, Y. C. Tay, and Yi Irvette Zhang*. A 3 level Cache Miss Model for a Nonvolatile Extension to Transcendent Memory . Proceedings of the 6th International Conference on Cloud Computing Technology and Science, pages 218–225, 2014.

# References

7. *Jinho Hwang, Wei Zhang, Ron C. Chiang, Timothy Wood, and H. Howie Huang*. UniCache: Hypervisor Managed Data Storage in RAM and Flash. Proceedings of the 7th International Conference on Cloud Computing, pages 216–223, 2014.

8. *Jiaxin, Jiwi, YouYou*. A high performance file system for NVM. Proceedings of the 11th European conference on computer systems 2016.

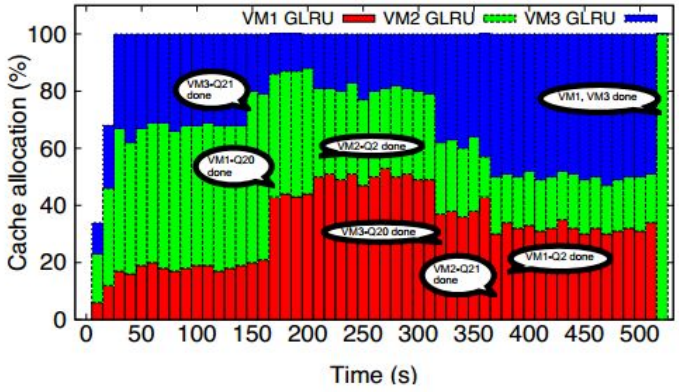
9. *Subramanya, Amitabha, Narayanan, Jifi*. Data tiering in heterogeneous memory system. Proceedings of the 11th European conference on computer systems 2016.

10. *Dan, Fang, Hong, Jei, Wei, Zheng*. Improving hybrid FTL by fully exploiting SSD parallelism. Transaction on architecture and code optimisation 2015

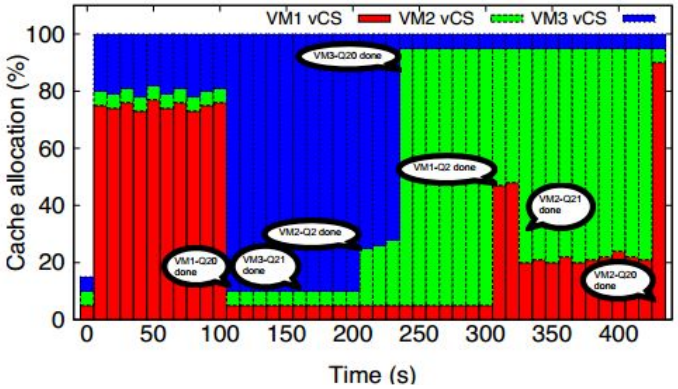
11. *Xian Chen, Wenzhi Chen, Member, IEEE, Zhongyong Lu, Peng Long, Shuiqiao Yang, and Zonghui Wang*. A Duplication-Aware SSD-Based Cache Architecture for Primary Storage in Virtualization Environment. IEEE Journal.

# Thank You

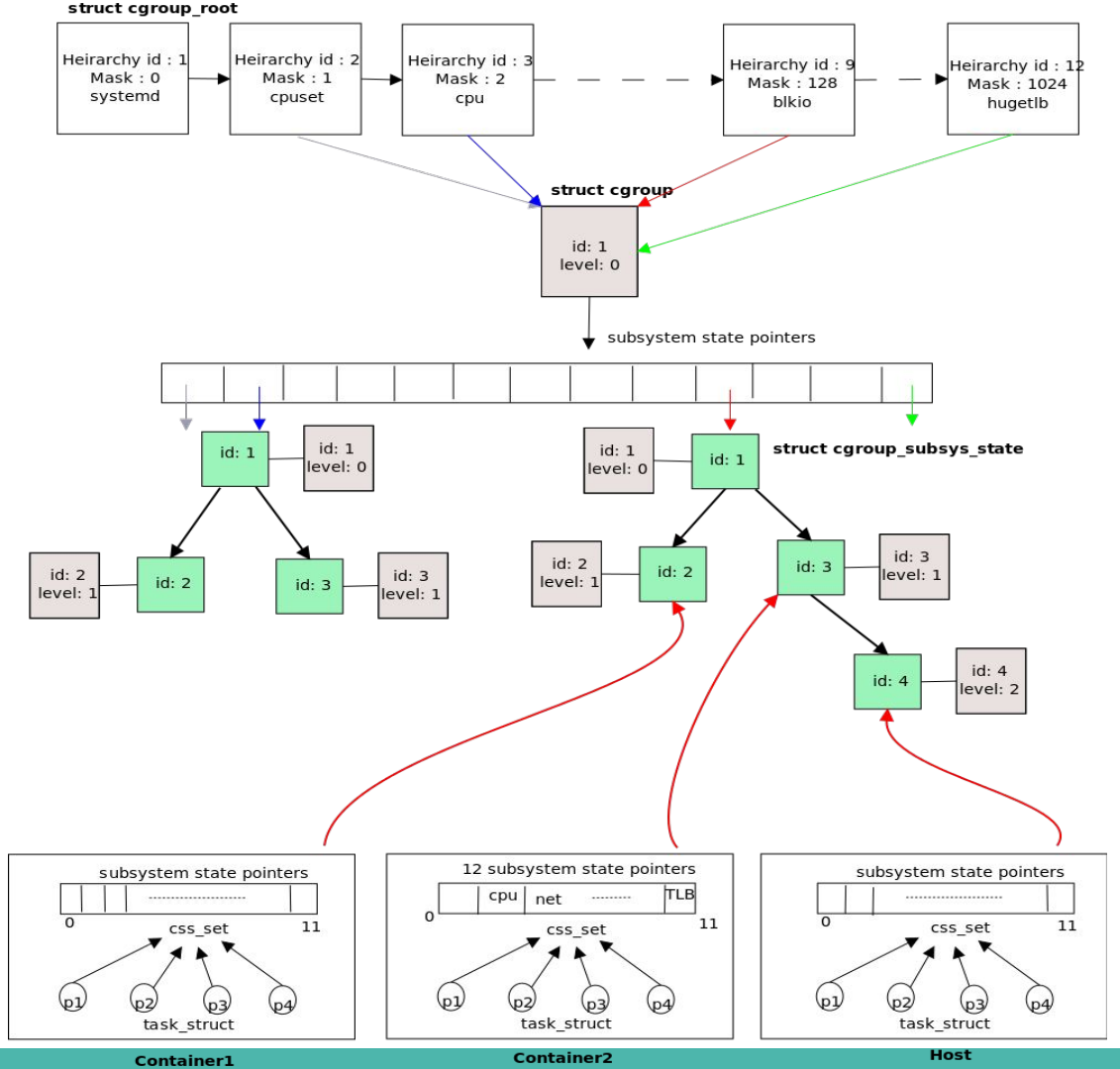
# Experiment - Static vs Dynamic




(a) G-LRU cache policy



# Cgroups


struct cgroup_root
Heirarchy id : 1
Mask : 0
systemd
Heirarchy id : 2
Mask : 1
cpuset
Heirarchy id : 3
Mask : 2
cpu
Heirarchy id : 9
Mask : 128
blkio
Heirarchy id : 12
Mask : 1024
hugetlb
struct cgroup
id: 1
level: 0
subsystem state pointers
id: 1
id: 1
level: 0
id: 2
level: 1
id: 2
id: 3
id: 3
level: 1
id: 1
level: 0
id: 1
struct cgroup_subsys_state
id: 2
level: 1
id: 2
id: 3
id: 3
level: 1
id: 4
id: 4
level: 2
subsystem state pointers
0
css_set
11
p1
p2
p3
p4
task_struct
Container1
12 subsystem state pointers
cpu
net
TLB
0
css_set
11
p1
p2
p3
p4
task_struct
Container2
subsystem state pointers
0
css_set
11
p1
p2
p3
p4
task_struct
Host

# Cache wastage

Container1

Request rate :10 blocks per sec

Less temporal locality

No block is referenced again

Container 2

Request rate : 6 blocks per second

More temporal locality

Same 6 blocks are referred

10 cache blocks

# Cache wastage

Container1

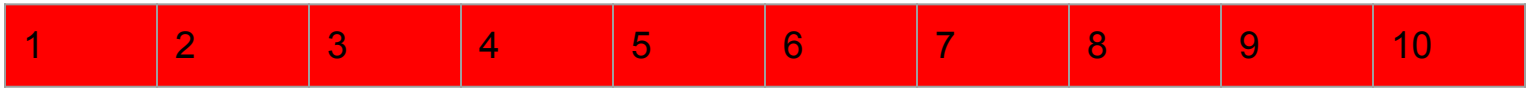
Request rate :10 blocks per sec

Less temporal locality

Container 2

Request rate : 6 blocks per second

More temporal locality




10 cache blocks

# Cache wastage

Container1

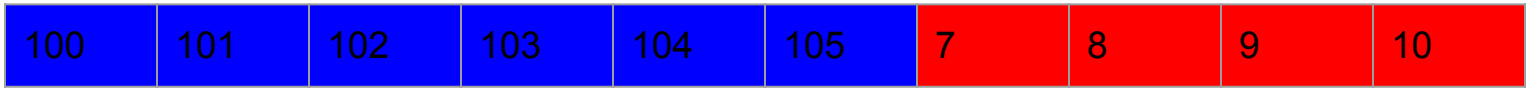
Request rate :10 blocks per sec

Less temporal locality

Container 2

Request rate : 6 blocks per second

More temporal locality




10 cache blocks

# Cache wastage

Container1

Request rate :10 blocks per sec

Less temporal locality

Container 2

Request rate : 6 blocks per second

More temporal locality


10 cache blocks

# Cache wastage

Container1

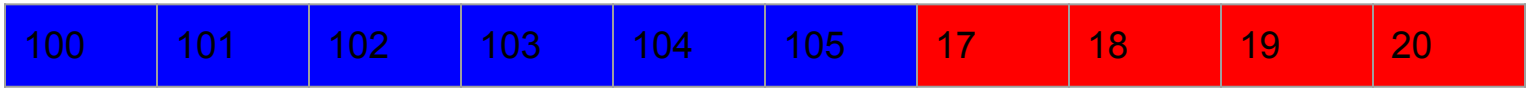
Request rate :10 blocks per sec

Less temporal locality

Container 2

Request rate : 6 blocks per second

More temporal locality




10 cache blocks

Experiment on Correctness

It can be verified that the module is storing the blocks when they are read from disk. And when blocks are not present in page cache same blocks are there in SSD. So SSD is acting as cache to disk.

Adhere to Cache limit

Cache limit is adhered since as number of blocks exceed cache size they are replaced using FIFO replacement policy.