

Differentiated SSD Caching for Container-based Hosting

Presented by

Shyamli Rao (153050009)

Under Guidance of Prof. Purushottam Kulkarni

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

27-06-2017

Problem Context

- Containers are emerging technology
- Nowadays SSDs are widely used as cache

Container

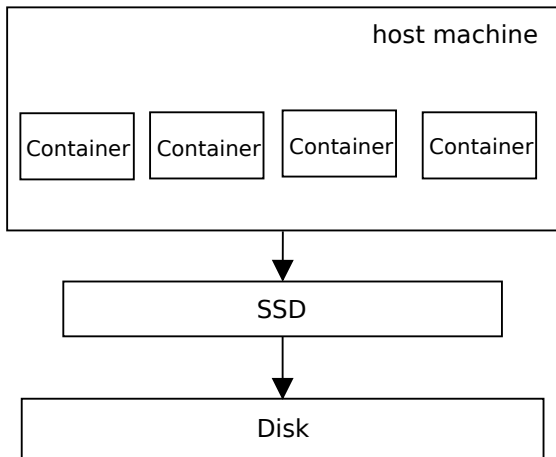
- Set of processes
- Isolated
- Resource Control

SSD as second level cache


Table: Comparison of memory technologies [7]


	DRAM	Flash	HDD
Read latency	10-50 ns	25 μ s	5ms
Write latency	10-50 ns	200 μ s	5ms
Cost	\$2 per gigabyte	\$0.20 per gigabyte	\$0.03 per gigabyte

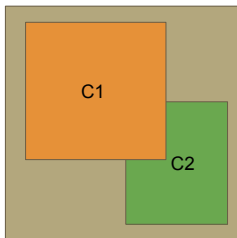
SSD as cache in Container-based Hosting



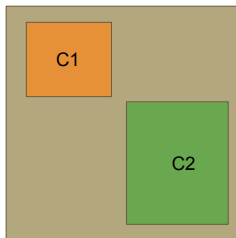
Motivation for Partitioning : Cache Wastage

C1 : Container 1's cache occupancy in SSD 

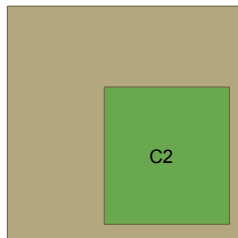
C2 : Container 2's cache occupancy in SSD 



Current scenario



Desired scenario



Problem Statement

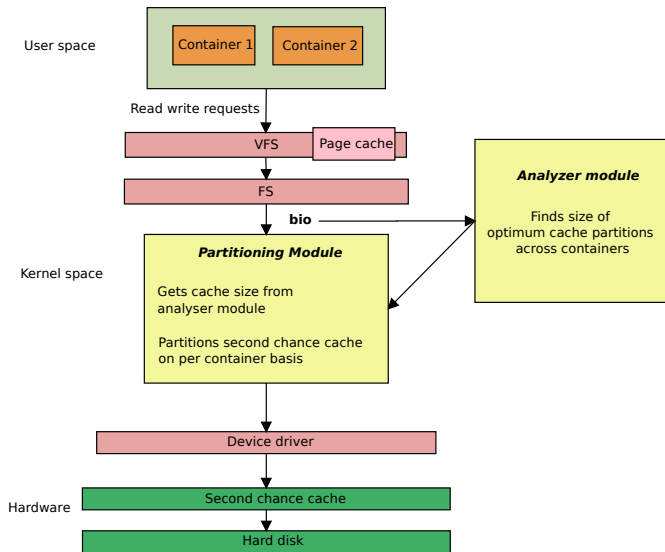
- Engineering a new design for :
 - Partitioning SSD
 - Deciding policy

Related Work

Table: Existing Solutions for VMs

Deciding policy	Fairness	SLA	Minimizing overall latency	Partitioning
ExTmem [5]	×	×	✓	×
vcacheShare [1]	×	×	✓	Dynamic
3-level [6]	✓	✓	×	Dynamic
Unicache [2]	✓	×	✓	Dynamic
Centaur [4]	✓	✓	✓	Dynamic

Architecture



Implementation of Partitioner

- Existing implementations for exposing SSD as cache :
 - dmccache
 - flashcache
- Flashcache
 - Set associative caching
 - Evictions per set

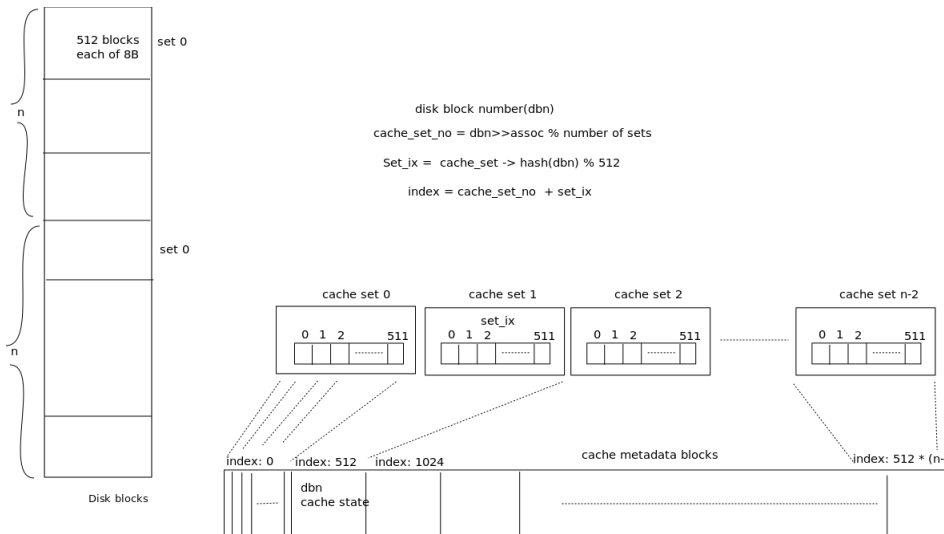
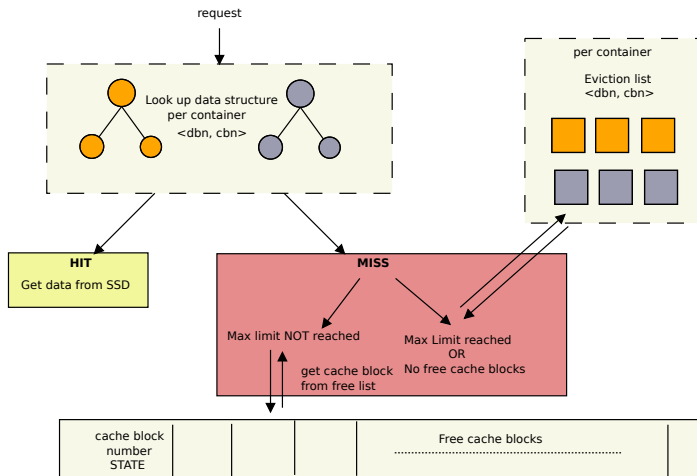


Figure: Flashcache Architecture

Partitioner module

- Global list of unused cache
- Per container :
 - lookup tree
 - eviction list
 - cache size limit

IO request handling



Analyser Module

- Maintains per container block access list
- Cache sizing can be based on
 - Working Set size [PARDA] [3]
 - Priority
 - SLA
 - Fairness
- After certain time interval calculates reuse distances
- Gets the optimal cache size of a container

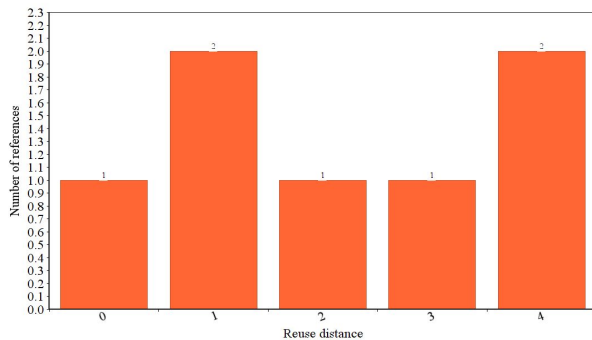
Reuse distance algorithm

Reuse distance of reference 'B' is number of unique references since the previous access to 'B'

time	0	1	2	3	4	5	6	7	8	9	10	11
references	d	a	c	b	c	c	e	b	a	d	a	c
Reuse distance	infi	infi	infi	infi	1	0	infi	2	3	4	1	4

Reuse distance algorithm

Hits = Sum(number of references) whose reuse distance $<$ cache size



Cache size	hits	Misses (12 - hits)
1	1	11
2	3	9
3	4	8
4	5	7
5	7	5
6	7	5

Experimental Evaluation

Correctness Verification

Question : Is SSD caching the blocks correctly when there is no contention on SSD resource?

Setup :

- Page cache - 256 MB
- SSD - 2 GB
- File size - 512 MB (131072 blocks)

Table: Correctness verification without evictions

	Container requests	Hits	Misses	Evictions
1st Run	131072	0	131072	0
2nd Run	131072	131072	0	0

Correctness Verification

Question : Can SSD cache the blocks correctly given a specific SSD size?
Is eviction working?

Setup :

- Page cache - 256 MB
- SSD - 500 MB (128000 blocks)
- File size - 512 MB (131072 blocks)

Table: Correctness verification with evictions

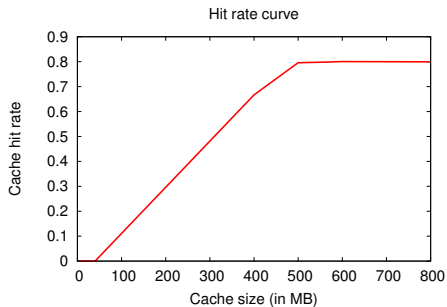
	Container requests	Hits	Misses	Evictions
1st Run	131072	0	131072	3072
2nd Run	131072	0	131072	131072

Effect of cache size on performance

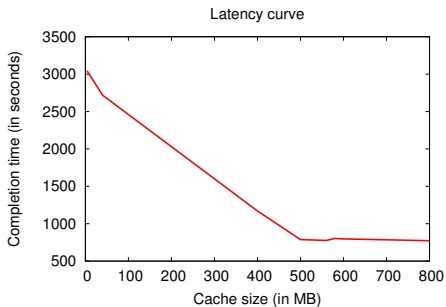
Question : How does SSD cache size effect cache hit ratio and completion time?

- Setup
 - Number of Containers - 1
 - Page Cache - 256 MB
- Workload
 - Reading file of working set size = 768 MB
- Metric
 - Completion time
 - Cache hit ratio
- Parameter - SSD Cache size

Effect of cache size on performance



(a) Hit rate curve



(b) Latency curve

Unified vs Partitioned SSD cache

Effect of varying IOPS

Question : How does performance of one container gets affected by IO rate of other container which is running in parallel?

■ Setup

- Number of containers - 2
- Page cache - 256 MB
- SSD - 1GB

■ Parameter - IO rate

■ Metric

- Cache hit ratio
- Completion time

■ Workload

- Read requests Gaussian access pattern with $\sigma = 50000$ on 10 GB file
- Read requests Gaussian access pattern with $\sigma = 500000$ on 10 GB file

Effect of varying IOPS : Workload

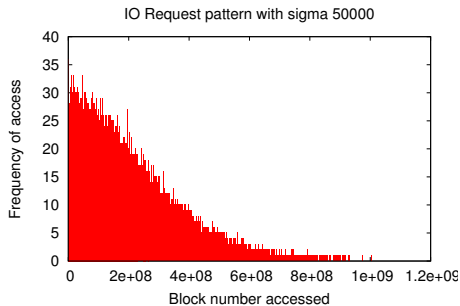


Figure: $\sigma = 50000$

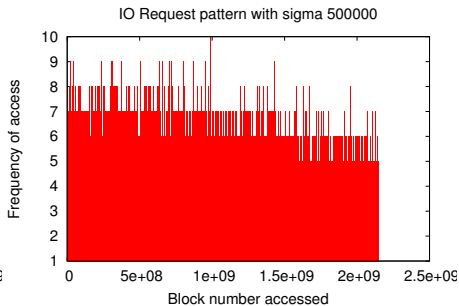


Figure: $\sigma = 500000$

Effect of varying IOPS

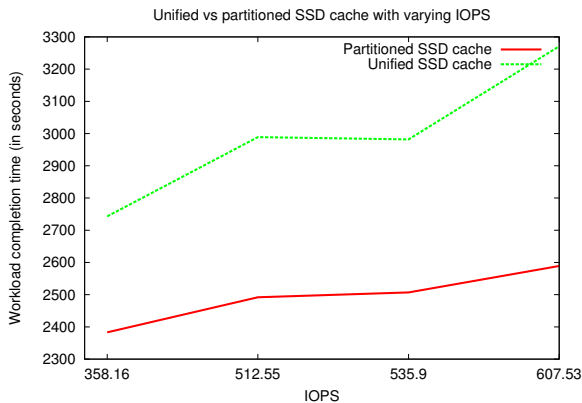
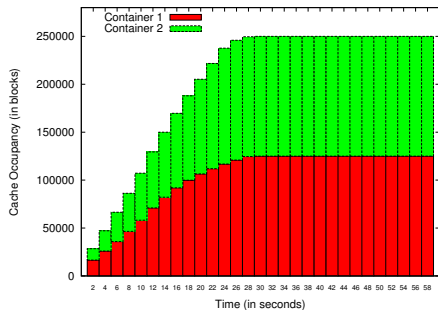
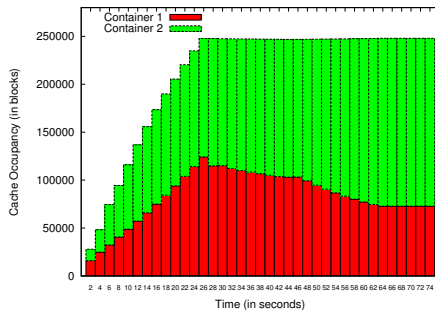


Figure: Effect on performance of Container 1 with varying IO rate of Container 2

Effect of varying IOPS : Cache occupancy



(a) Partitioned SSD cache



(b) Unified SSD cache

Figure: Cache occupancy at IOPS = 607.53

Effect of varying IOPS : Cache hit ratio

Table: Cache hit ratio

	Partitioned cache hit ratio		Unified cache hit ratio		Overall Hit rate	
IOPS	C1	C2	C1	C2	Partitioned	Unified
358.16	78.26	29.48	68.19	33.96	53.87	51.075
512.55	79.58	29.46	53.25	34.41	54.52	43.83
535.90	79.54	29.47	48.06	33.89	54.50	40.97
607.53	79.58	28.89	41.29	34.15	54.23	37.72

Effect of varying workloads

Question : How does performance of one container gets affected by working set size and access pattern of other container running on same host?

- Setup

- Number of containers - 2
- Page cache - 256 MB
- SSD - 1GB

- Parameter - Workload access pattern

- Metric

- Cache hit ratio
- Completion time

- Workload

- Read requests Gaussian access pattern with $\sigma = 50000$ on 10 GB file

Effect of varying workloads : Workloads

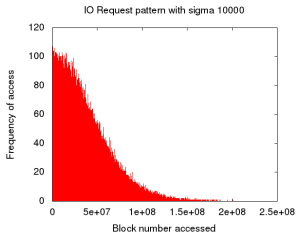


Figure: $\sigma = 10000$

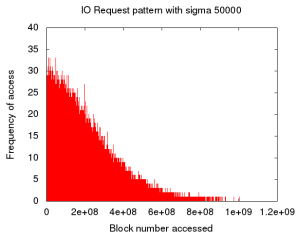


Figure: $\sigma = 50000$

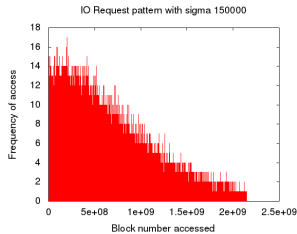


Figure: $\sigma = 150000$

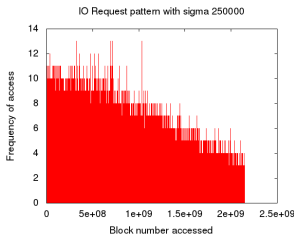


Figure: $\sigma = 250000$

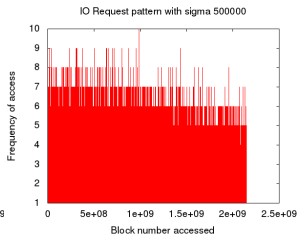


Figure: $\sigma = 500000$

Effect of varying workloads

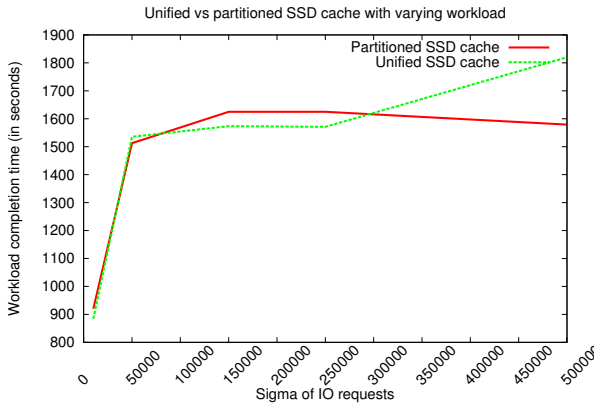
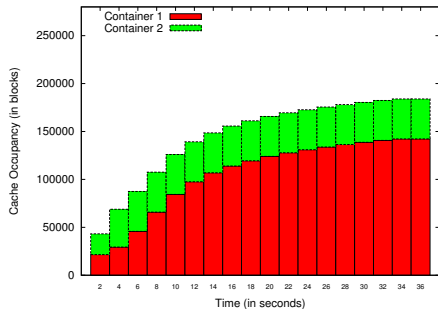
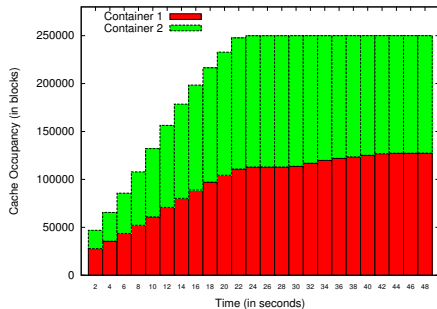


Figure: Effect on performance with varying working set size

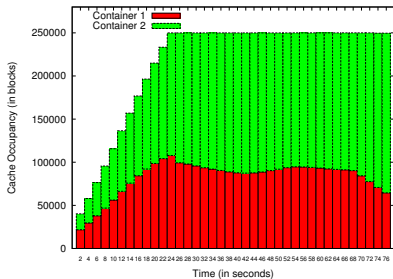
Effect of varying workloads : Cache occupancy



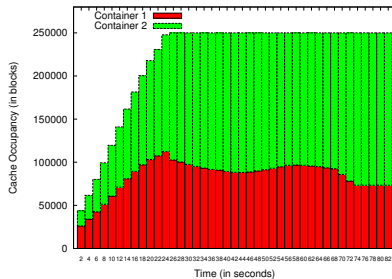
(a) IO Request pattern with $\sigma = 10000$



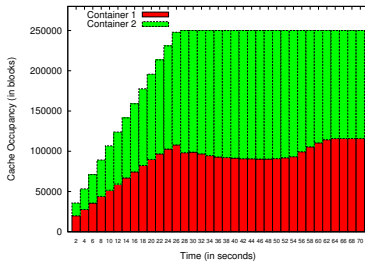
(b) IO Request pattern with $\sigma = 50000$



(c) $\sigma = 150000$



(d) $\sigma = 250000$



(e) $\sigma = 500000$

Effect of varying workloads : Cache hit ratio

Table: Cache hit ratio of both the containers

Sigma	Partitioned cache hit ratio		Unified cache hit ratio		Overall Hit rate	
	C1	C2	C1	C2	Partitioned	Unified
10000	80.18	in page cache	79.39	in page cache	80.18	79.39
50000	79.62	79.42	78.18	78.80	79.52	78.49
150000	79.59	33.84	66.02	51.66	56.71	58.84
250000	79.44	29.91	65.87	44.95	54.67	55.41
500000	79.36	29.5	58.47	32.92	54.43	45.695

Real Workload

Question : How does performance get affected with statically partitioned cache and unified cache?

- Setup

- Number of containers - 4
- Page cache - 256 MB
- SSD - 1GB

- Metric

- Cache hit ratio
- Throughput

- Workload

- Webserver : 700MB file is accessed by 10 threads simultaneously
- Random reads : 2GB file read by 32 threads simultaneously

Real Workload : Cache occupancy

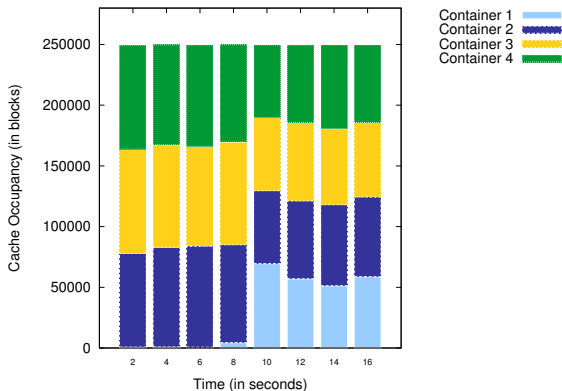


Figure: Cache occupancy of unified cache

Real Workload : Cache hit ratio

Table: Cache hit ratio of workloads

Workload	Hit ratio Unified	Hit ratio partitioned
Webserver	15.9	86.94
Random reads	12.75	10.75
Random reads	12.32	10.98
Random reads	12.97	10.60

Real Workload : Throughput comparison

Table: Throughput comparison

Workload	Throughput Unified (MB/s)	Throughput Partitioned (MB/s)
Webserver	12.8	28.5
Random reads	11.3	10.3
Random reads	12.6	10.1
Random reads	12.8	10.6

Dynamic partitioning : Correctness Verification

Question : Is the module predicting the optimal cache size correctly?

- Setup

- Number of containers - 1
- Page cache - 256 MB
- Epoch time - 120 seconds

- Workload

- Reading a file of size 10 GB with Gaussian distribution with working set size of 142276 blocks

Dynamic partitioning

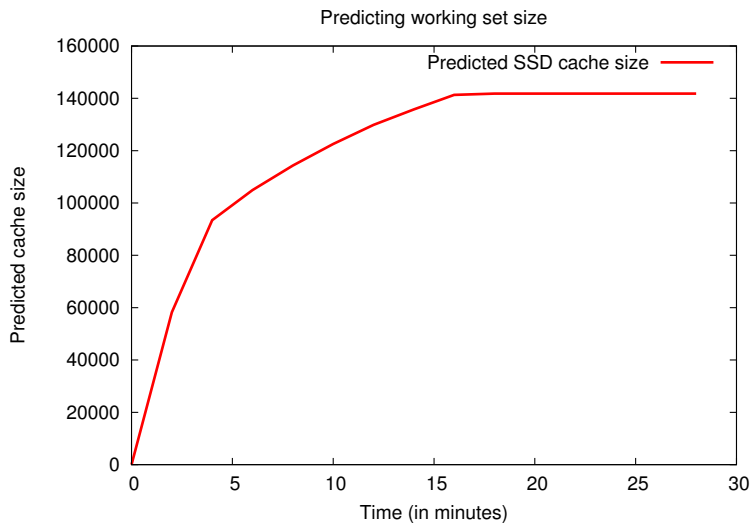


Figure: Predicting cache size dynamically

Future Work

- Analyser module to support limited SSD cache size
- Experiments on performance of dynamic partitioning
- Resizing of cache can be based on page cache access patterns

Thank you

References I



Meng Fei, Zhou Li, Ma Xiaosong, Uttamchandani Sandeep, and Liu Deng.

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.



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.

References II



Qingda Lu Qingpeng Niu, James Dinan and P. Sadayappan.
PARDA: A Fast Parallel Reuse Distance Analysis Algorithm.
Proceedings of the 22nd international conference on Parallel architectures and compilation techniques, pages 103–112, 2013.



R. Rangaswami R. Koller, A. J. Mashtizadeh.
Centaur: HostSide SSD Caching for Storage Performance Control.
Proceedings of Autonomic Computing, 2015 IEEE International Conference on cloud computing, pages 966–973, 2015.



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.

References III



Vimalraj Venkatesan, Wei Qingsong, Y. C. Tay, and Yi Irvette Zhang.
A 3level 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.



Carl A. Waldspurger.

A study of application performance with non-volatile main memory .

*Proceedings of the 31th symposium on Mass Storage Systems and
Technologies (MSST)*, 2015.