Adaptive Memory Management Frameworks for Derivative Clouds

M.Tech. Thesis Presentation

Prashanth 153050095

Advisor:
Prof. Purushottam Kulkarni

Cloud provider architecture

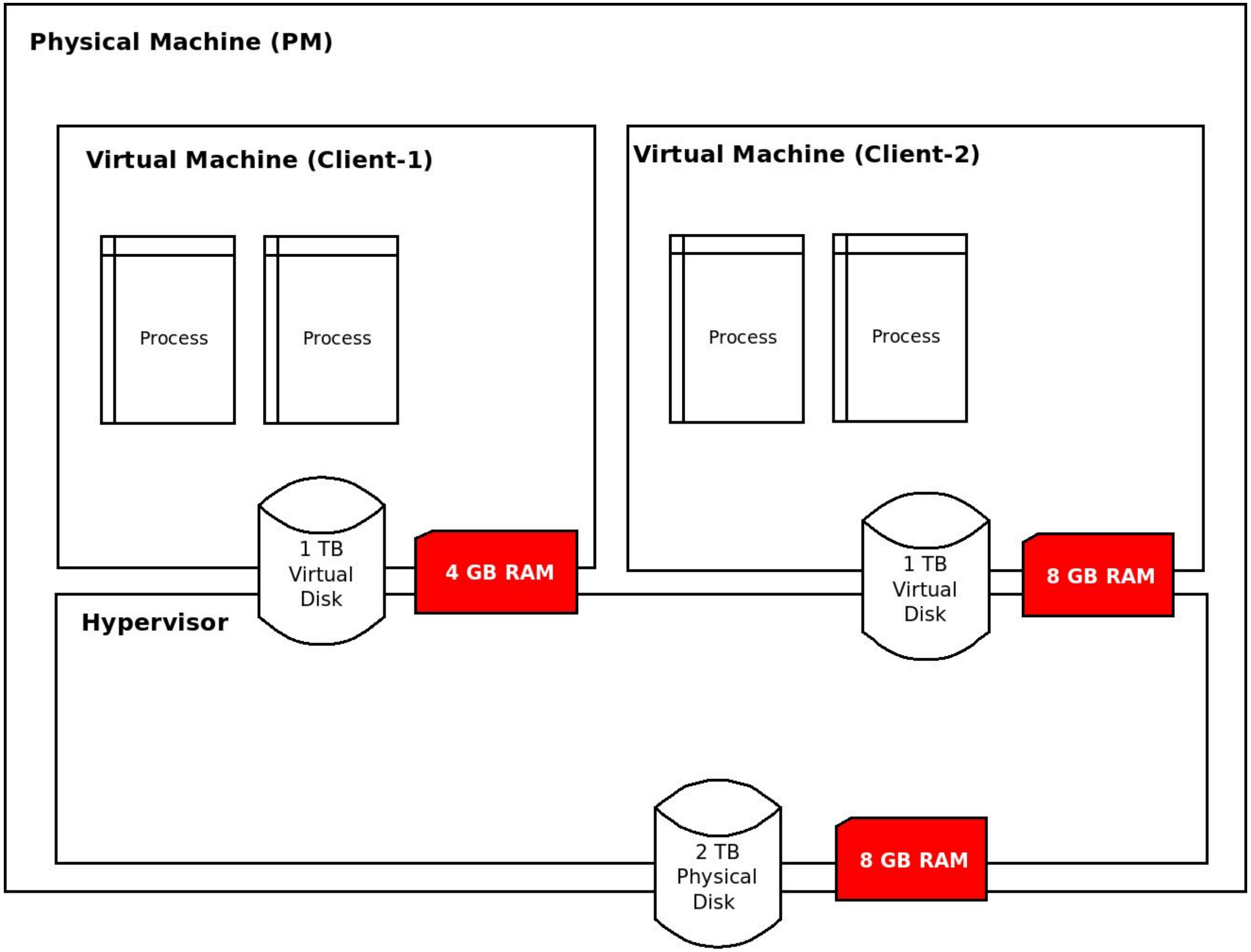
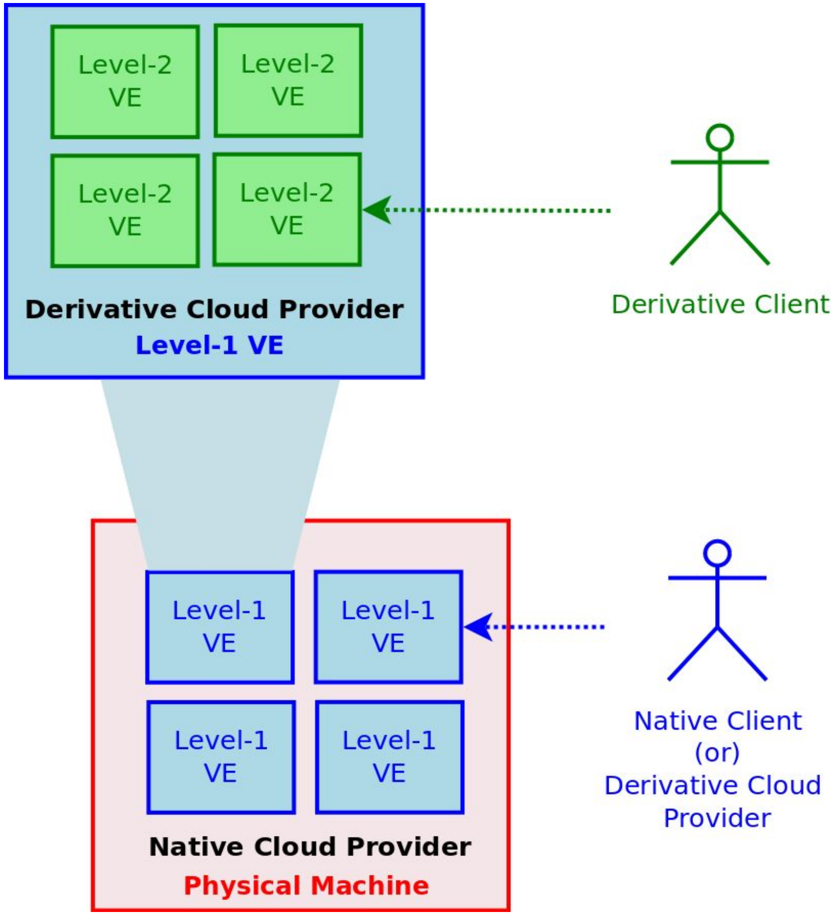


Fig-1: Cloud providers provisioning clients using VMs

Key Points

1. Provision clients on VMs
2. Map SLA requirements to VM resources
3. **Overcommit resources for cost benefits**

Derivative cloud environment



- **VE (Virtual Environment):** Virtual Machine (VM) or Container
- Level-1 VE - VM
  Level-2 VE - Container
- **2 control centers = 2 levels of overcommitment**
- [Spotcheck EuroSys '15]
  [Heroku PAAS provider]
  [Google cloud platform]

Fig-2: Comparison between native and derivative cloud environment

MTP stage-1 overview

- Surveyed existing literature on Linux memory management for containers

- Negative impacts of existing memory controller in an overcommitted setup
    1. Native cloud
    2. Derivative cloud

- Proposed a **new differentiated controller** to enforce a wider range of policies

- New control knob - **Relative shares**

- Our work was accepted at *ICDCS '17* in the poster track
  ***Joint work with Chandra Prakash*

Caching in a cloud setup

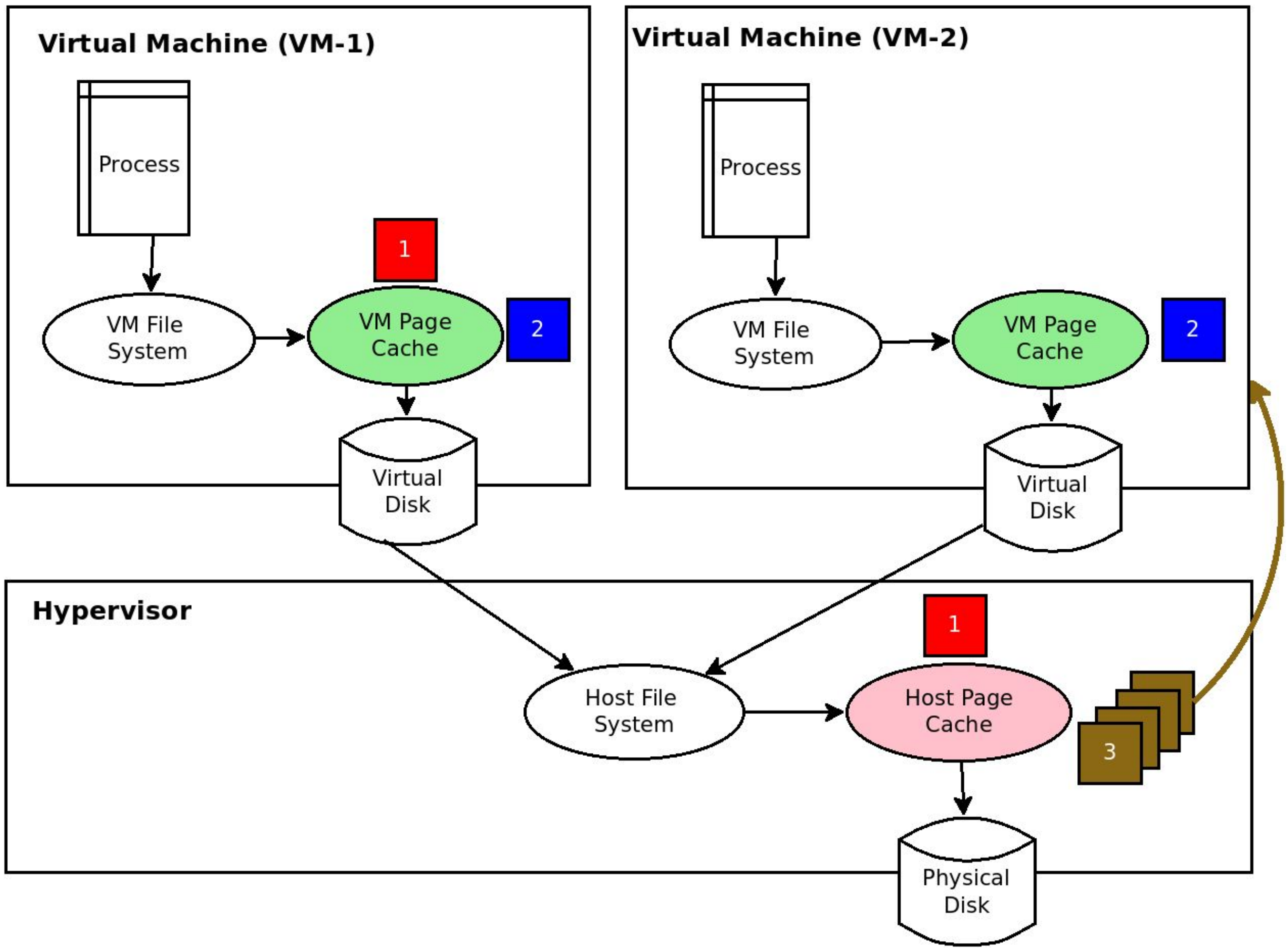


Fig-3: File I/O in traditional hypervisor based cloud setup

Drawbacks

1. Multiple copies of same page at host and VM
2. Multiple copies between VMs
3. Flooding of host page cache by a particular VM

Hypervisor managed cache

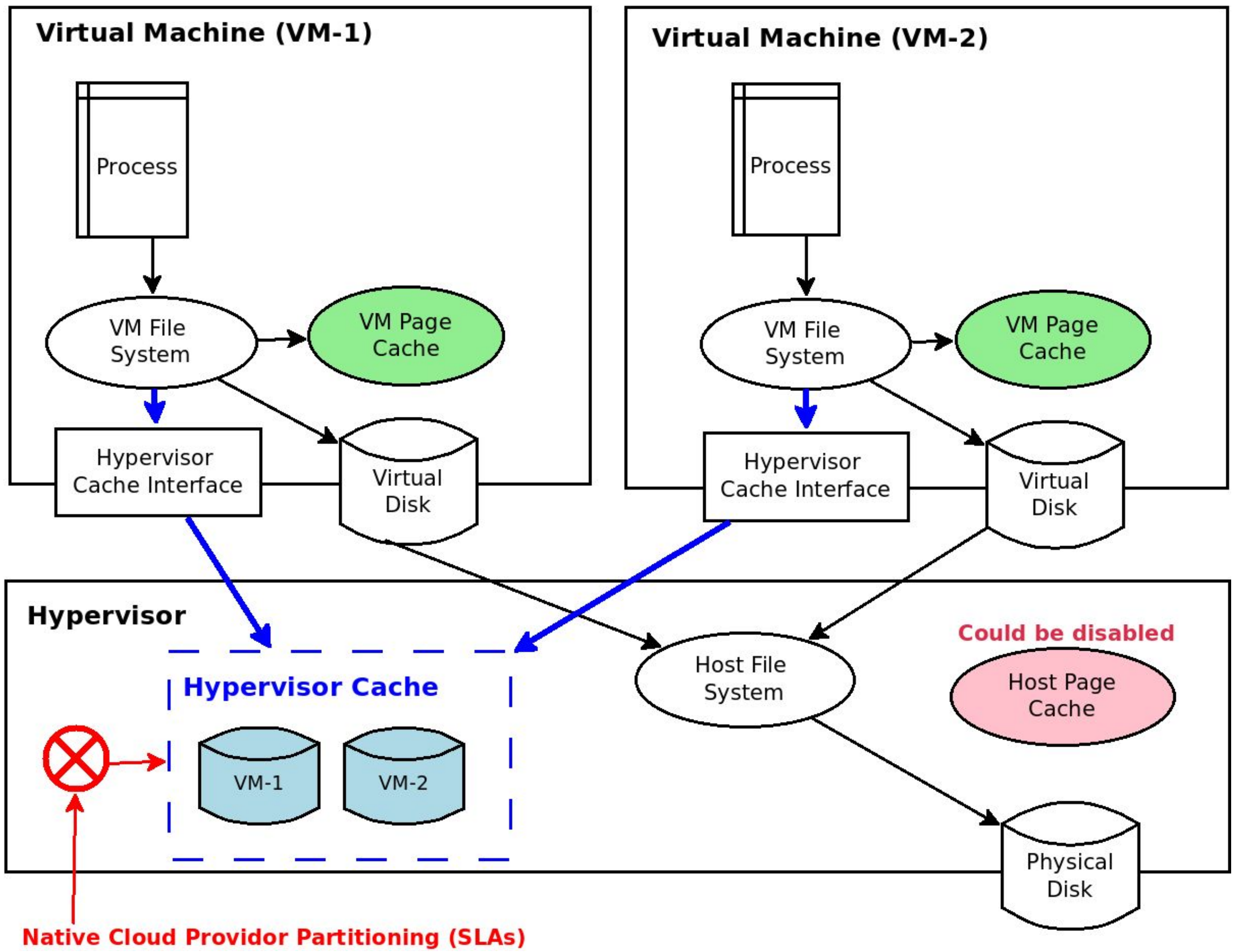


Fig-4: Hypervisor managed cache architecture

Key Points

1. **Each VM is treated as a single application**
2. Partitioning per VM based on SLAs
3. Existing works address this issue - [SDC SoCC '15] [Centaur ICAC '15]
4. Exclusive cache

Marks the controller

What are caches backed by ?

- Caches could be backed by
  1. Memory (RAM)
  2. SSD
  3. NVMs etc.

- We could even combined them to form multi-level or hybrid cache designs - [Ex-tmem HPCC ‘14]

Problem statement

Initial motivation

- ❏ To develop a caching framework to support derivative clouds

Problem description

- ❏ To build a memory management framework to satisfy application objectives (SLA) in a derivative cloud environment
- ❏ Supports both
    1. Memory provisioning
    2. Cache partitioning (Multi-level caches)

Related works

- **Memory management**

  [Ballooning SIGOPS '02], [Singleton HPDC '12] , [Overdriver SIGPLAN '11], **[Ex-tmem HPCC '14]**

- **Hypervisor managed cache**

  [Comparative analysis MASCOTS '14] [Mercury MSST '12] [S-CAVE PACT '13]

- **Hypervisor Cache partitioning**

  [SDC SoCC '15] [Centaur ICAC '15]

- **Derivative clouds**

  [Spotcheck Eurosys '15]

Memory management framework for derivative clouds

Caching framework for derivative clouds

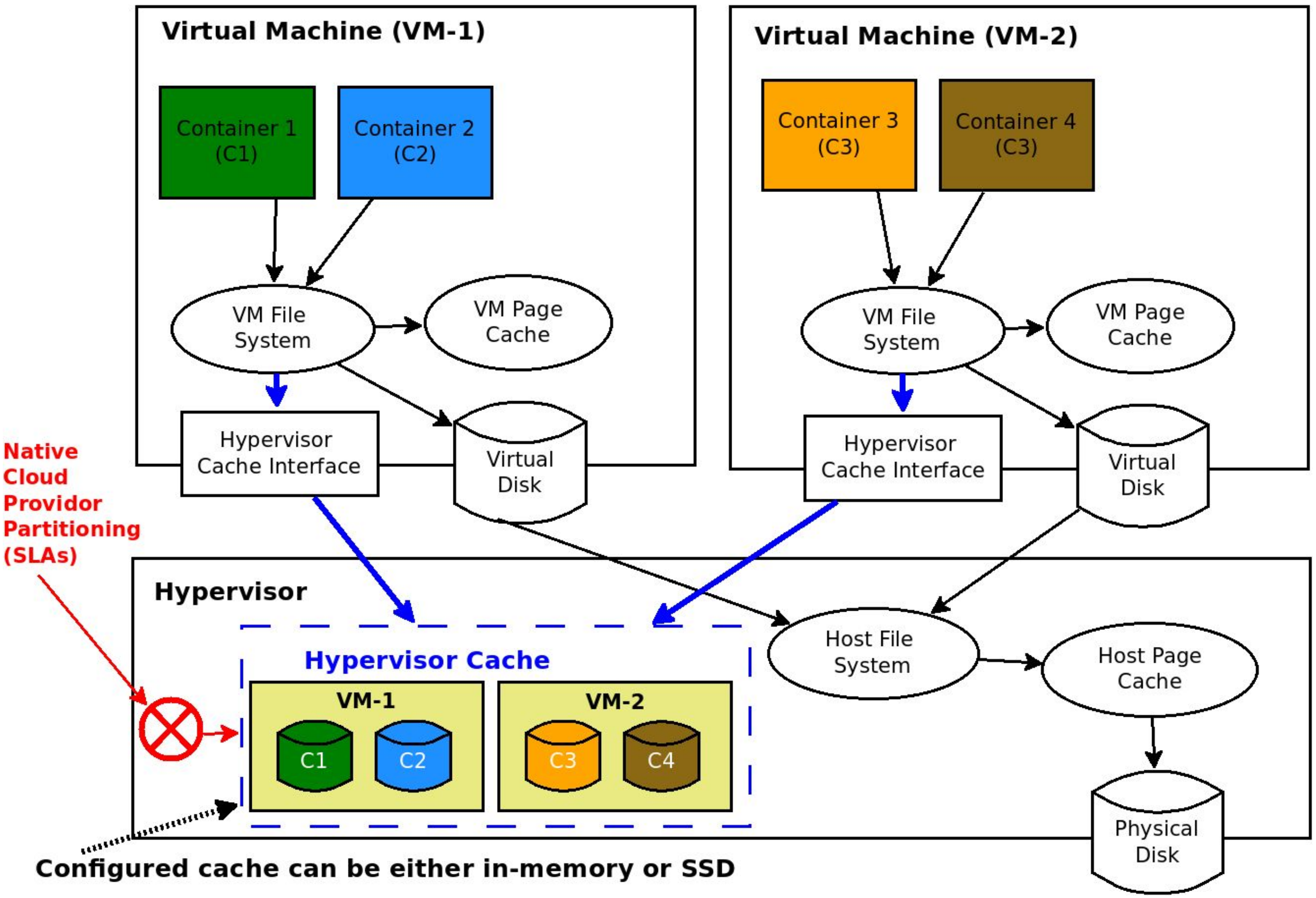


Fig-5: Background work - Cache partitioning framework for derivative clouds

Key Points

Each container is treated as one application and provisioned

Native cloud provider is the controlling entity

The initial work revolved around this idea [DoubleDecker ‘16]

Drawbacks of traditional single-level cache partitioning frameworks

Experimental testbed

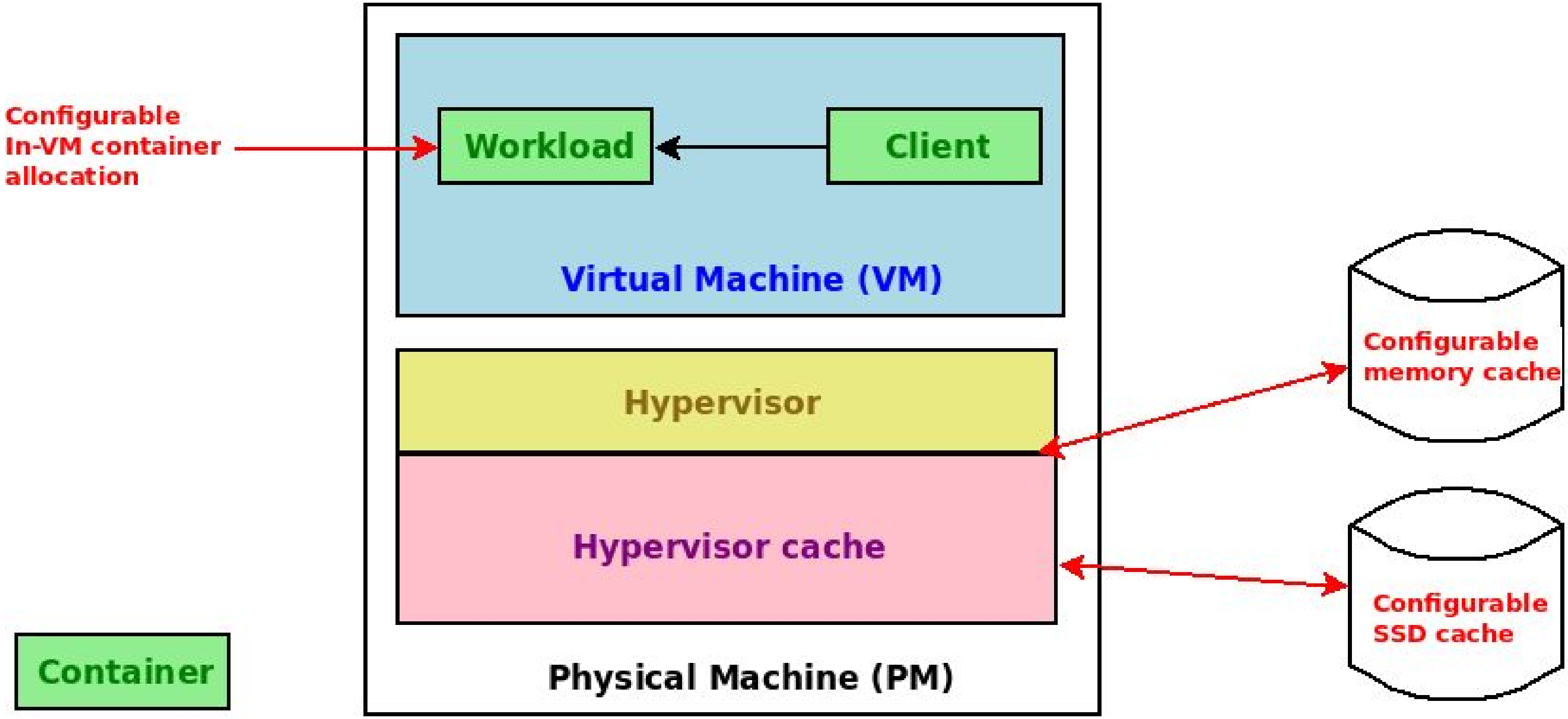


Fig-6: Experimental testbed for establishing drawbacks of traditional cache partitioning frameworks

Parameters and metrics

Parameters varied

1. Memory requirement (WSS)
2. Container memory limit
3. Cache limit (Memory or SSD)
4. Workloads
5. Number of containers

Metrics collected

1. Application metrics (throughput and latency)
2. Container memory usage
3. Cache stats (Memory or SSD)

    usage, inserts, requests, flushes, evicts, promotion, demotion

Workloads used

Synthetic CAT	A self generated workload that *cat* a file onto */dev/null*
Web server (Filebench)	[Filebench] is a synthetic workload to emulate a web server
MongoDB	[MongoDB] is NoSQL database application
Redis	[Redis] is an in-memory key value store
MySQL	[MySQL] is a database workload that uses anonymous pages to configure its own user-space data cache
YCSB Client	Yahoo Cloud Server Benchmark [YCSB project] as the benchmark to generate the clients evaluate key-value stores

Effects of memory allocations on application performance

- 4 application of two types,
    1. File backed applications - Web server, MongoDB
    2. Anonymous memory applications - Redis, MySQL (uses own cache)

- Each application WSS = 2 GB

- Split their provisioning into ratios of **In-memory : Cache**

Ratio	Actual allocation	Meaning
8 : 0	2 GB : 0 GB	All memory provisioned at VM
4 : 4	1 GB : 1 GB	Half in-VM and other in cache
1 : 7	1.75 GB : 0.25 GB	Nearly all on cache

Table-1: To illustrate allocation ratios meaning

Effects of memory allocations on application performance

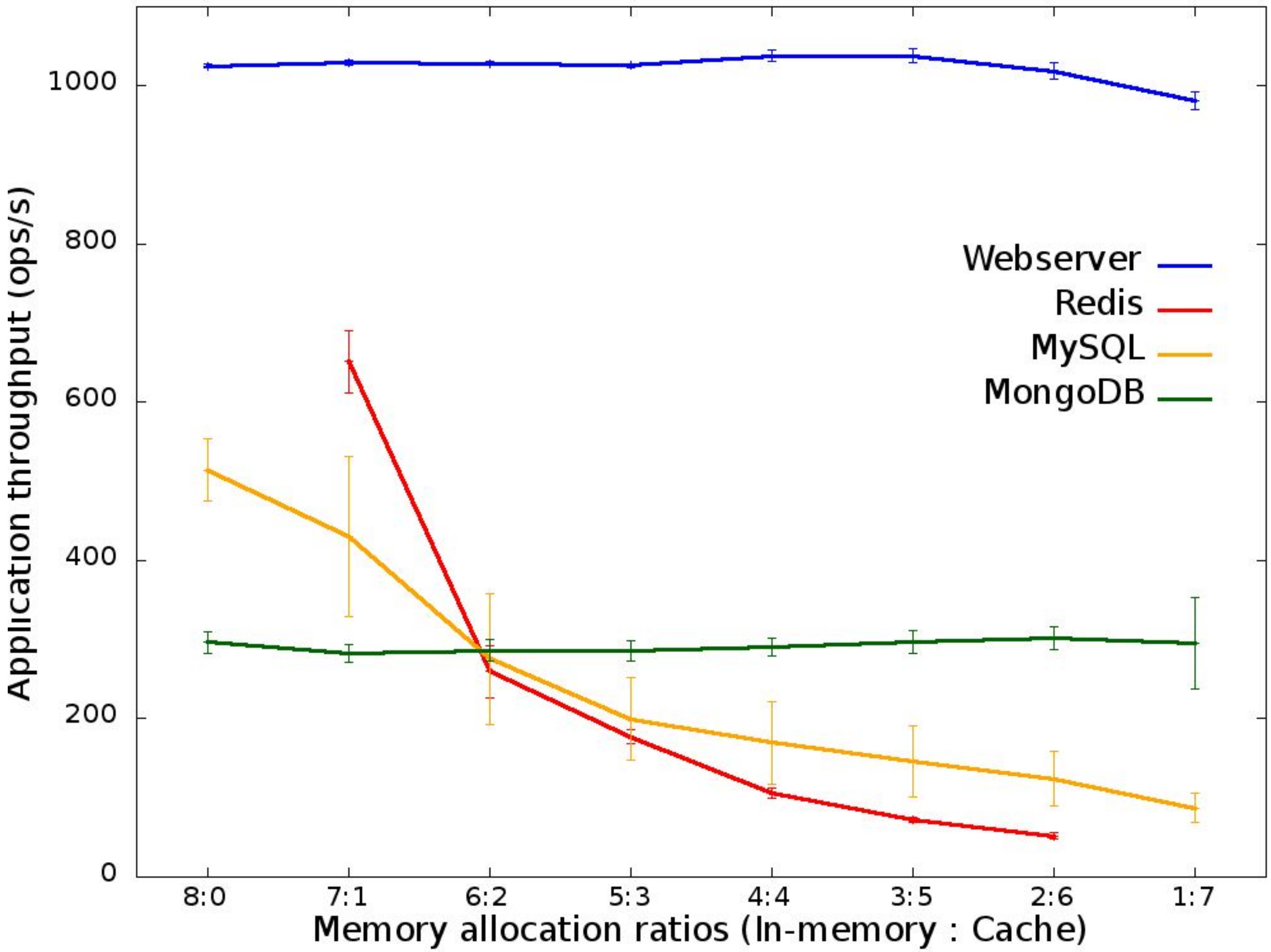


Fig-7: Memory allocation ratios and how it affects application performance

Inferences

- File-backed workloads like Web Server and MongoDB can be at either places
- **Anonymous memory workloads like Redis, MySQL require in-memory allocations**

Motivation for a hybrid cache configuration

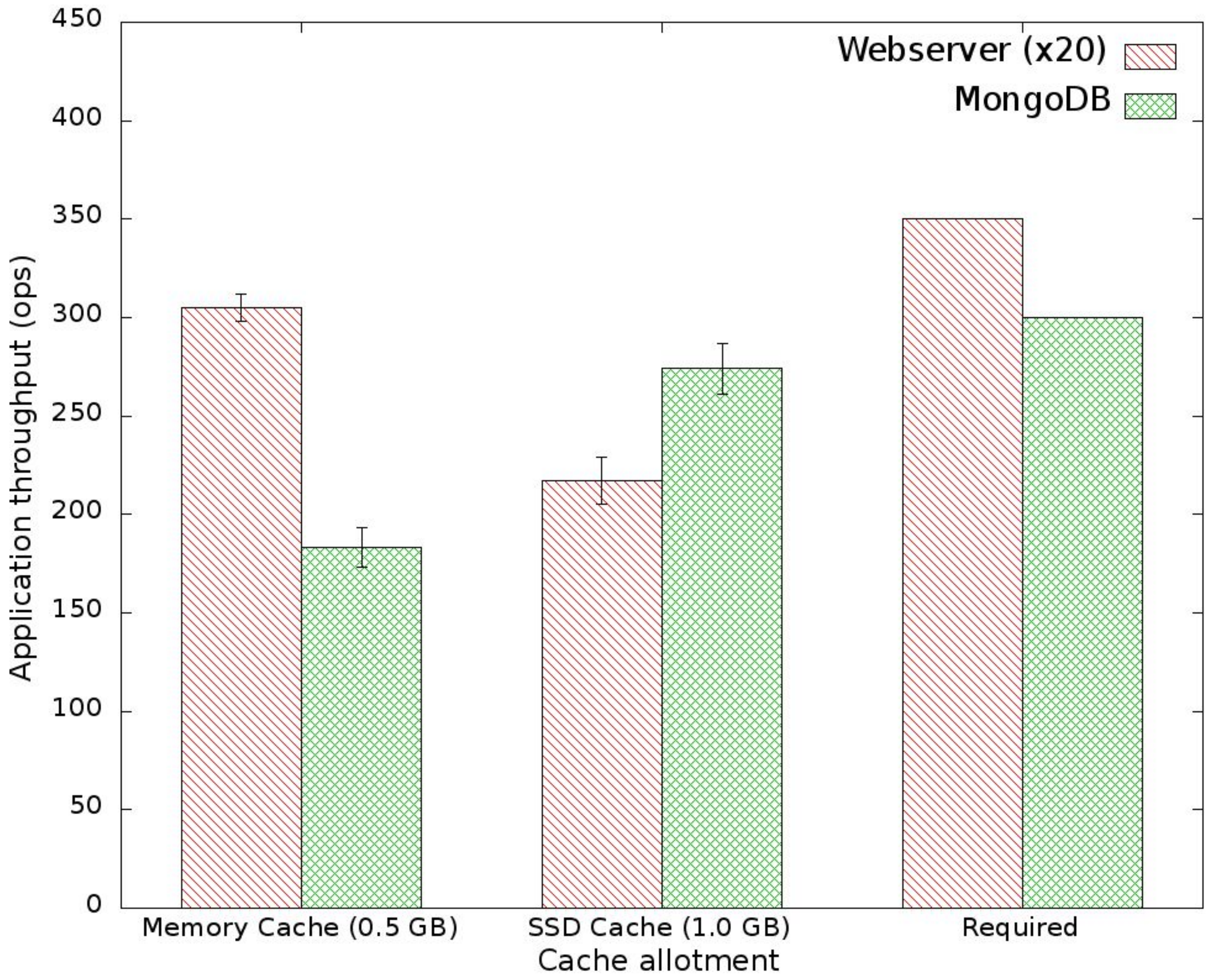


Fig-8: Application throughput achieved by provisioning at different cache levels

Scenario

Cache available at the two levels,

Memory	0.5 GB
SSD	1.0 GB

Required throughput not satisfied by either cache levels

Hybrid setup is desirable

Requirements of desired system

- **Native provider -** Configure memory and cache allocations to all VMs
- **Derivative provider -** Configure memory and cache allocations for all containers executing within a VM
- Hybrid multi-level configurable caches
  1. Level-1 cache (Memory)
  2. Level-2 cache (SSD)
- Spillover mechanism - Exceeds in Level-1 spilled over to the Level-2 cache
- Resource conserving

Decentralized memory management framework

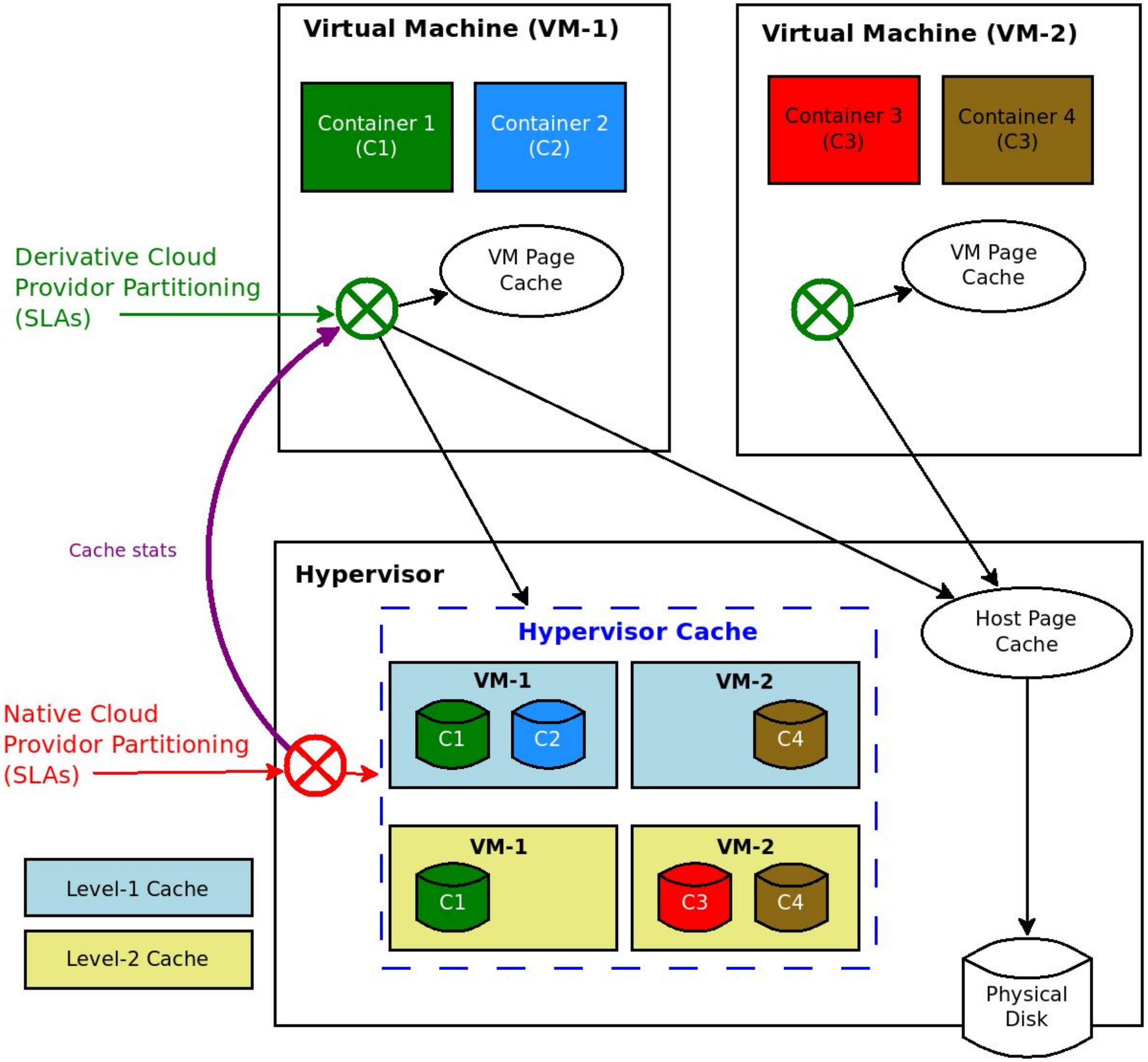


Fig-9: Architecture for decentralized memory management framework for derivative clouds

Key points

Two control centers,

1. Native provider
2. Derivative provider

Each of whom can configure 3 knobs,

1. Memory allocation
2. L1 (Memory) cache size
3. L2 (SSD) cache size

Movement (or) eviction of objects

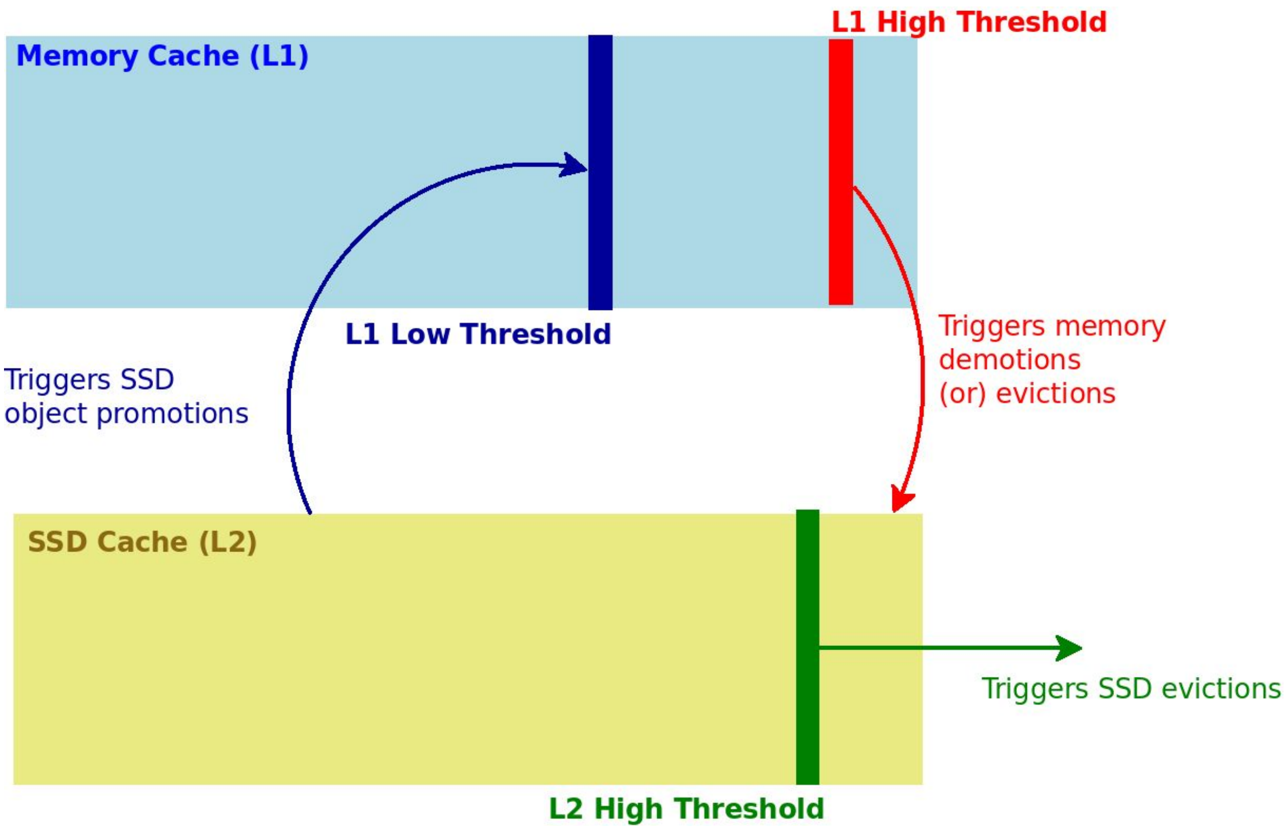


Fig-10: Design for movement of cache objects between levels

Key points

Eviction / movement algorithm,

1. First selects the VM most exceeding its allocation
2. Then selects the container within the VM which is most exceeding

Implementation details

Setup

- ❏ Linux + KVM + LXC
- ❏ T-MEM cache [Dan Magenheimer ‘09]
- ❏ Extended this to support hypervisor backed caches using memory and SSD

Crucial implementation components

1. Control knob - Relative weights
2. Cache store to accommodate both types of objects
3. Asynchronous kthreads to support movement / eviction of objects
4. Multi-level cache stats

Correctness of our implementation

Arithmetic validation of stats

$$CalculatedUsed = Puts + ObjectsMovedIn - (Gets + Flushes + ObjectsMovedOut)$$

- CalculatedUsed = ActualUsed

Movement of objects between both levels

- Synthetic cat workload to trigger movement of objects at both levels
- Compared estimated stats to actual stats values
- Our observations showed < 1% deviations from estimated stats

Effectiveness of hybrid design

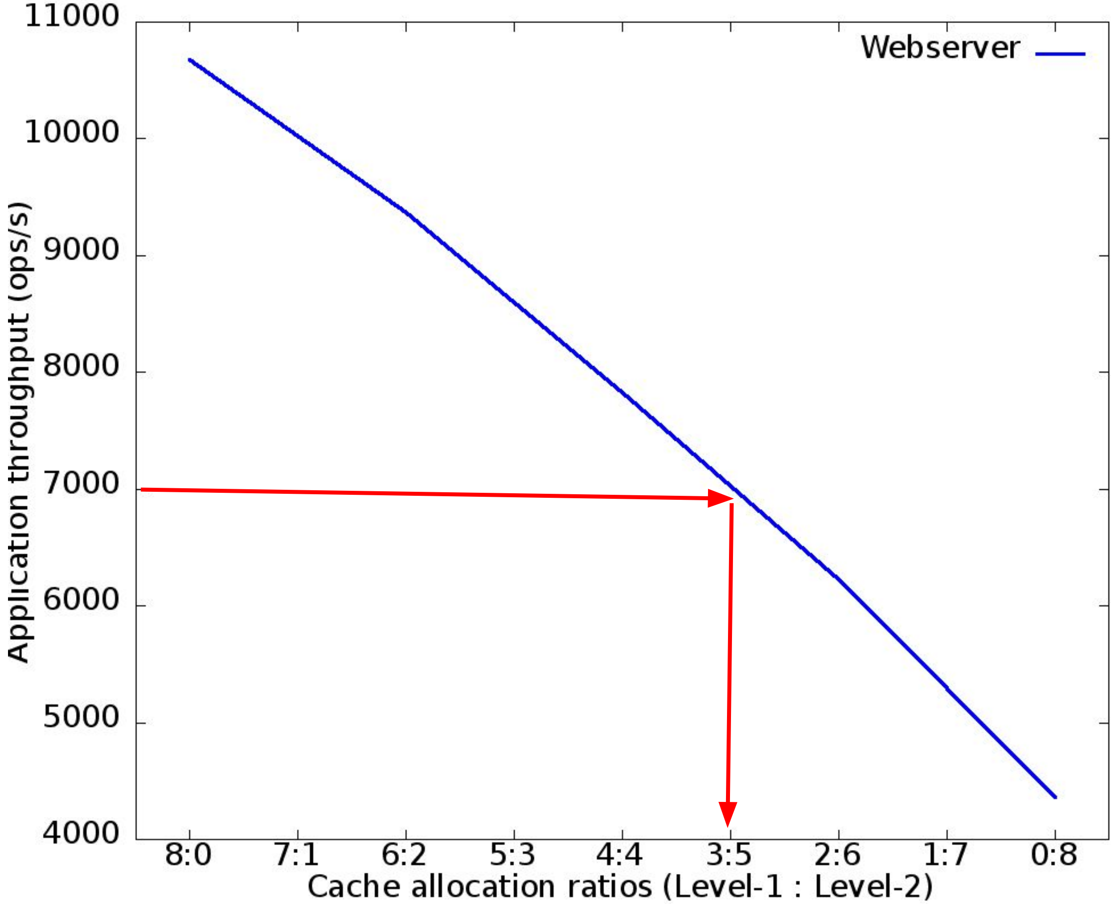


Fig-11: Cache requirement of 1024 MB divided between level-1 and level-2 cache in an hybrid setup for Web server

Min. required allocation		
Level-1	4	384 MB
Level-2	4	640 MB
Total	8	1024 MB

Effectiveness of hybrid design

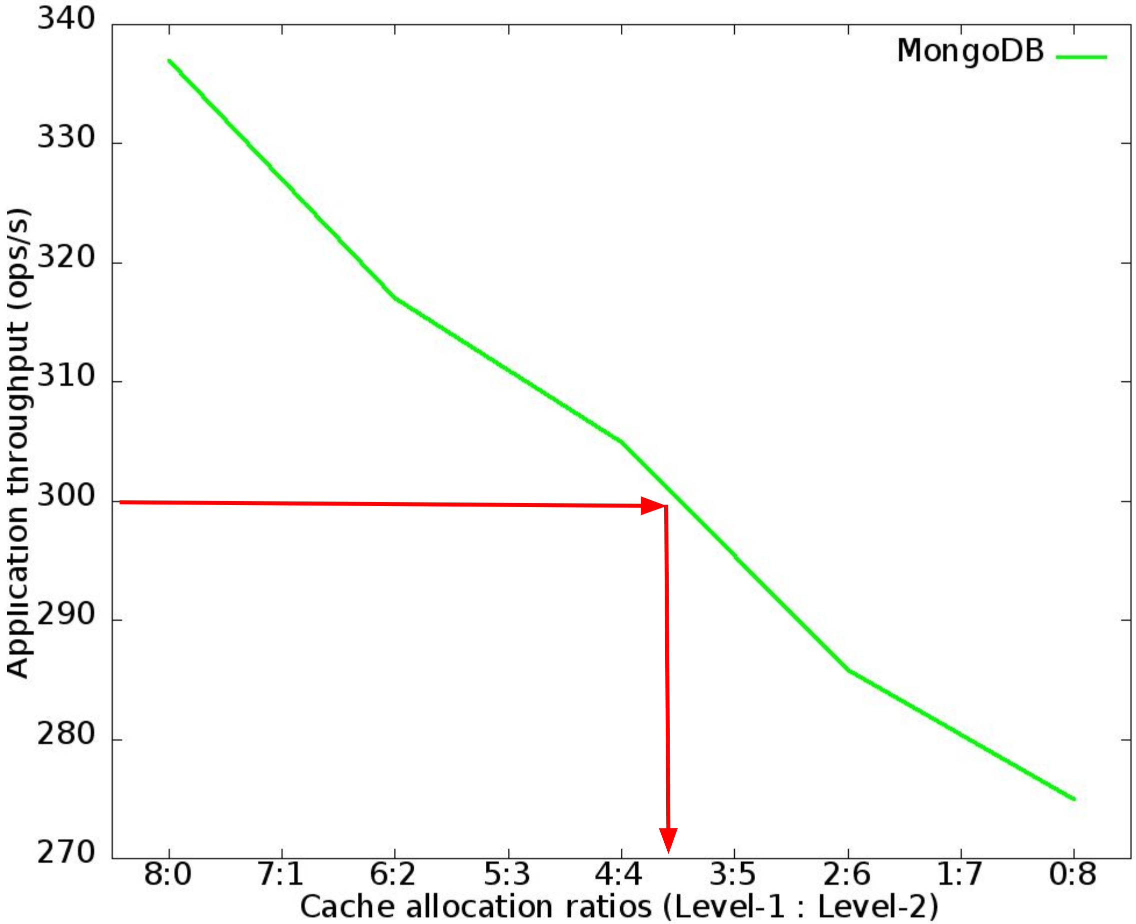
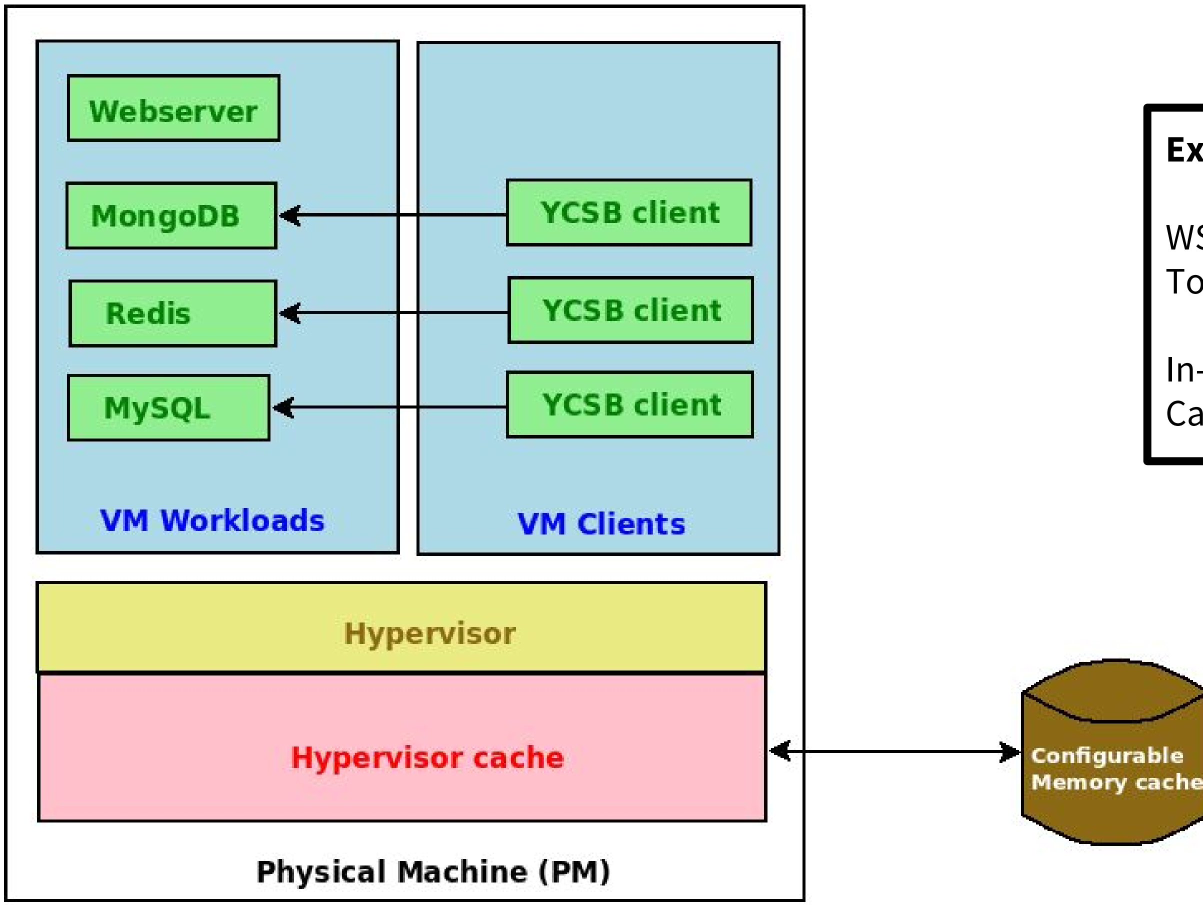


Fig-12: Cache requirement of 1024 MB divided between level-1 and level-2 cache in an hybrid setup for MongoDB

Min. required allocation

Level-1	4	512 MB
Level-2	4	512 MB
Total	8	1024 MB

Impact of decentralized controller



Experimental config.

WSS of each app.	2 GB
Total WSS	8 GB
In-VM memory size	6 GB
Cache size	2 GB

Fig-13: Experimental testbed used to show effectiveness of our decentralized controller over traditional cache partitioning frameworks

Three different configurations taken

All configurations have their **best cache partitioning schema**

1. **Unrestricted memory allocation**
   6 GB of in-VM memory is shared between all containers executing

2. **Uniform memory allocation**
   1.5 GB of in-VM memory is allocated to each container

3. **Best memory allocation (Our framework)**
   Most memory allocated to in-VM memory requirement workloads (Redis, MySQL), and cache allocated to other workloads

Impact of decentralized controller

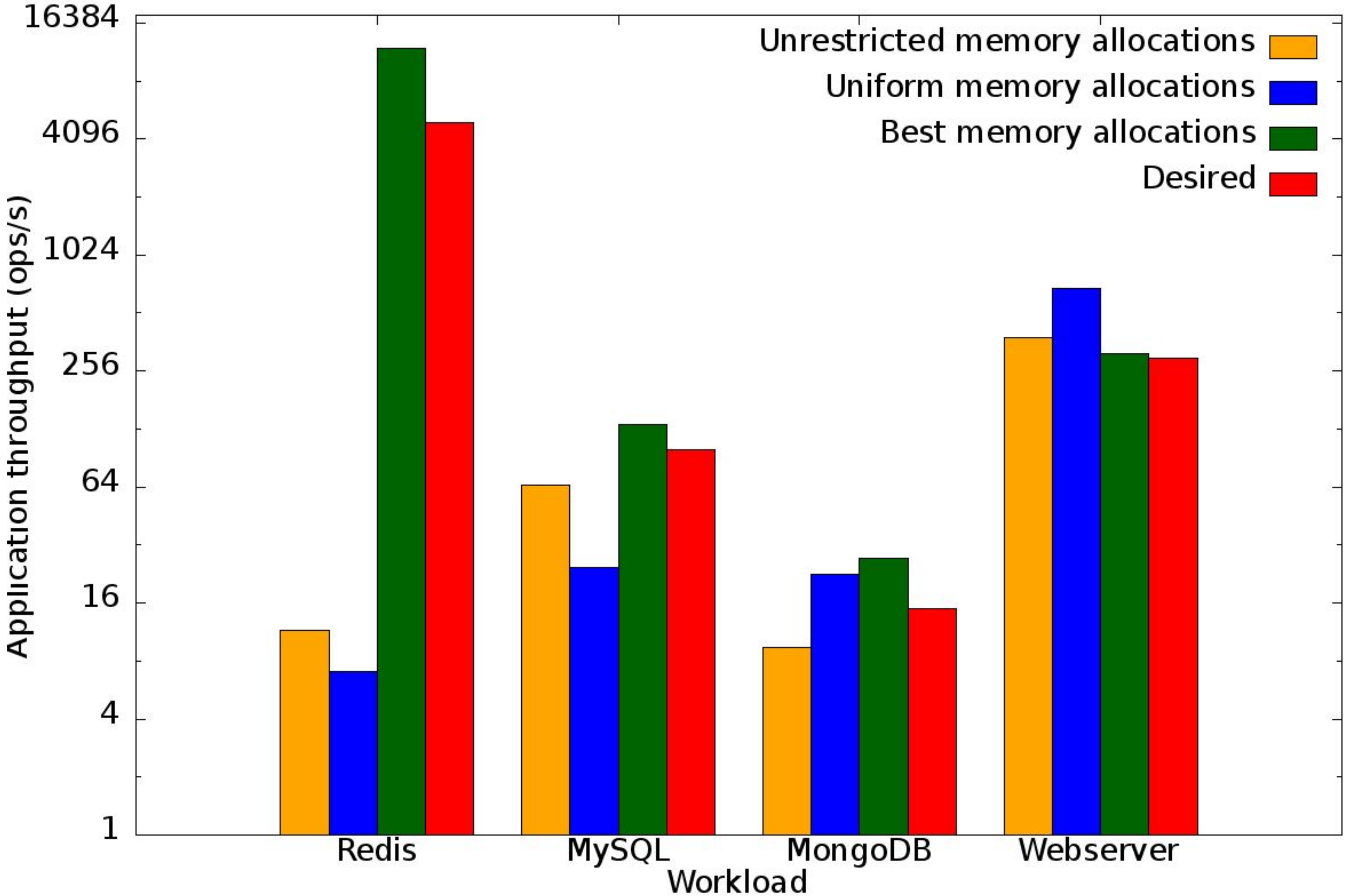


Fig-14: Impact of memory allocations to achieving application objectives with best cache partitioning in each case

Conclusion

- Proposed a decentralized memory management framework

- Multi-level configurable knobs
    1. Memory allocation
    2. Level-1 cache size
    3. Level-2 cache size

- Demonstrated the effectiveness of our framework in satisfying SLAs over traditional cache partitioning frameworks and single-level caches

- Decentralized memory management framework is in submission at *Middleware '17*
  ***Extension to work done by Debadatta Mishra*

Future Directions

- To build policies that can be enforced using our framework to support,
  1. Individual application level objectives
  2. Overall hypervisor level objective
  3. Per-VM level objectives

- Mapping of application objectives to spread over 3 levels of allocations
  Hint: map applications anonymous memory requirements onto the VM

- Explore other resources and how they are affected in an derivative cloud setup

References

[1] “Google cloud platform.” https://cloud.google.com/container-engine/

[2] “Heroku.” http://www.heroku.com.

[3] “Sap cloud platform.” https://cloudplatform.sap.com/index.html.

[4] “Getting your hands dirty with containers.” https://www.cse.iitb.ac.in/~prashanth/containers/seminar/manual.pdf.

[5] J. McKendrick, “Forbes business magazine: Is all-cloud computing inevitable? analysts suggest itis,” 2016.

[6] “Amazon elastic compute cloud.” https://aws.amazon.com/ec2/.

[7] T. Dörnemann, E. Juhnke, and B. Freisleben, “On-demand resource provisioning for bpel workflows using amazon’s elastic compute cloud,” in Cluster Computing and the Grid, 2009. CCGRID 09. 9th IEEE/ACM International Symposium on, pp. 140–147, IEEE, 2009.

[8] Z. Shen, S. Subbiah, X. Gu, and J. Wilkes, “Cloudscale: elastic resource scaling for multi-tenant cloud systems,” in Proceedings of the 2nd ACM Symposium on Cloud Computing, p. 5, ACM, 2011.

[9] R. Moreno-Vozmediano, R. S. Montero, and I. M. Llorente, “Elastic management of cluster-based services in the cloud,” in Proceedings of the 1st workshop on Automated control for datacenters and clouds, pp. 19–24, ACM, 2009.

[10] R. N. Calheiros, C. Vecchiola, D. Karunamoorthy, and R. Buyya, “The aneka platform and qos-driven resource provisioning for elastic applications on hybrid clouds,” Future Generation Computer Systems, vol. 28, no. 6, pp. 861–870, 2012.

References

[11] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and linux containers,” in Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On, pp. 171–172, IEEE, 2015.

[12] R. Morabito, J. Kjallman, and M. Komu, “Hypervisors vs. lightweight virtualization: a performance comparison,” in Cloud Engineering (IC2E), 2015 IEEE International Conference on, pp. 386–393,IEEE, 2015.

[13] K. Agarwal, B. Jain, and D. E. Porter, “Containing the hype,” in Proceedings of the 6th Asia-Pacific Workshop on Systems, p. 8, ACM, 2015.

[14] D. Beserra, E. D. Moreno, P. Takako Endo, J. Barreto, D. Sadok, and S. Fernandes, “Performance analysis of lxc for hpc environments,” in Complex, Intelligent, and Software Intensive Systems (CISIS), 2015 Ninth International Conference on, pp. 358–363, IEEE, 2015

[15] M. S. Rathore, M. Hidell, and P. Sjödin, “Kvm vs. lxc: comparing performance and isolation of hardware-assisted virtual routers,” American Journal of Networks and Communications, vol. 2, no. 4,pp. 88–96, 2013.

[16] P. Sharma, S. Lee, T. Guo, D. Irwin, and P. Shenoy, “Spotcheck: Designing a derivative iaas cloud on the spot market,” in Proceedings of the Tenth European Conference on Computer Systems, p. 16, ACM, 2015.

[17] “Redis in-memory key-value store.” http://redis.io/.

[18] “Memcached distributed in-memory key-value store.” https://memcached.org/.

[19] D. Mishra and P. Kulkarni, “Comparative analysis of page cache provisioning in virtualized environments,” in Modelling, Analysis & Simulation of Computer and Telecommunication Systems(MASCOTS), 2014 IEEE 22nd International Symposium on, pp. 213–222, IEEE, 2014

References

[20] R. Koller, A. J. Mashtizadeh, and R. Rangaswami, “Centaur: Host-side ssd caching for storage performance control,” in Autonomic Computing (ICAC), 2015 IEEE International Conference on,pp. 51–60, IEEE, 2015.

[21] I. Stefanovici, E. Thereska, G. O’Shea, B. Schroeder, H. Ballani, T. Karagiannis, A. Rowstron, and T. Talpey, “Software-defined caching: Managing caches in multi-tenant data centers,” in Proceedings Of the Sixth ACM Symposium on Cloud Computing, pp. 174–181, ACM, 2015.

[22] D. Mishra and P. Kulkarni, “Doubledecker: Differentiated hypervisor caching for derivative clouds,” 2016.

[23] “Stress workload generator.” http://people.seas.harvard.edu/~apw/stress/.

[24] “Mongodb.” https://docs.mongodb.com/v3.2/.

[25] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking cloud serving systems with ycsb,” in Proceedings of the 1st ACM symposium on Cloud computing, pp. 143–154,ACM, 2010.

[26] “Mysql database server.” https://www.mysql.com/.

[27] “Filebench.” https://github.com/filebench/filebench/wiki.

Thank You !

Any questions ?