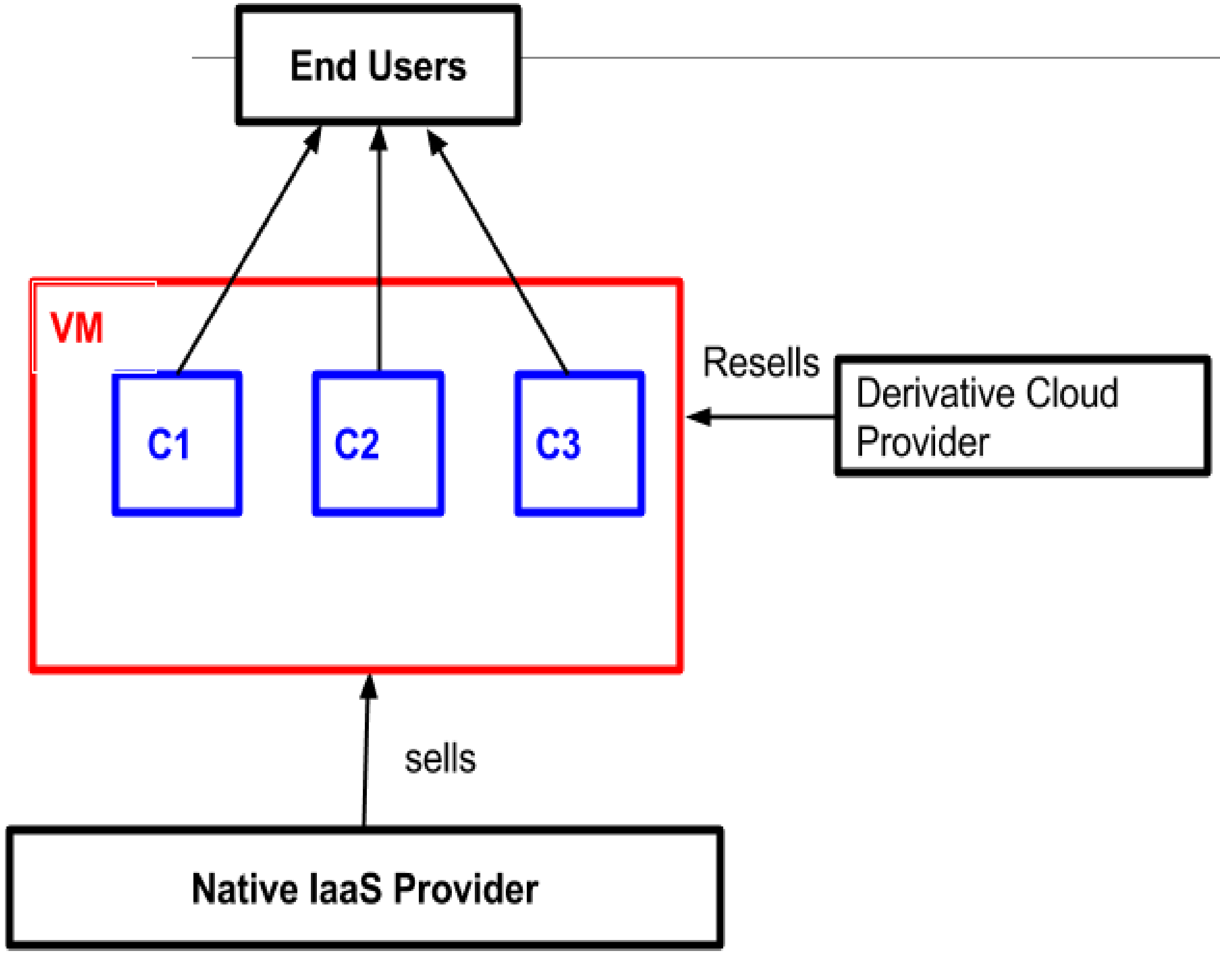
Hypervisor Cache Provisioning in Congnizance with Application Behaviour in Derivative Cloud

Kanika Pant

Master's Thesis

The Department of Computer Science and Engineering
IIT Bombay

Derivative Clouds



- Service on top service
- Multi-level control
- Resource management to satisfy application objectives

Hypervisor Managed Second chance cache

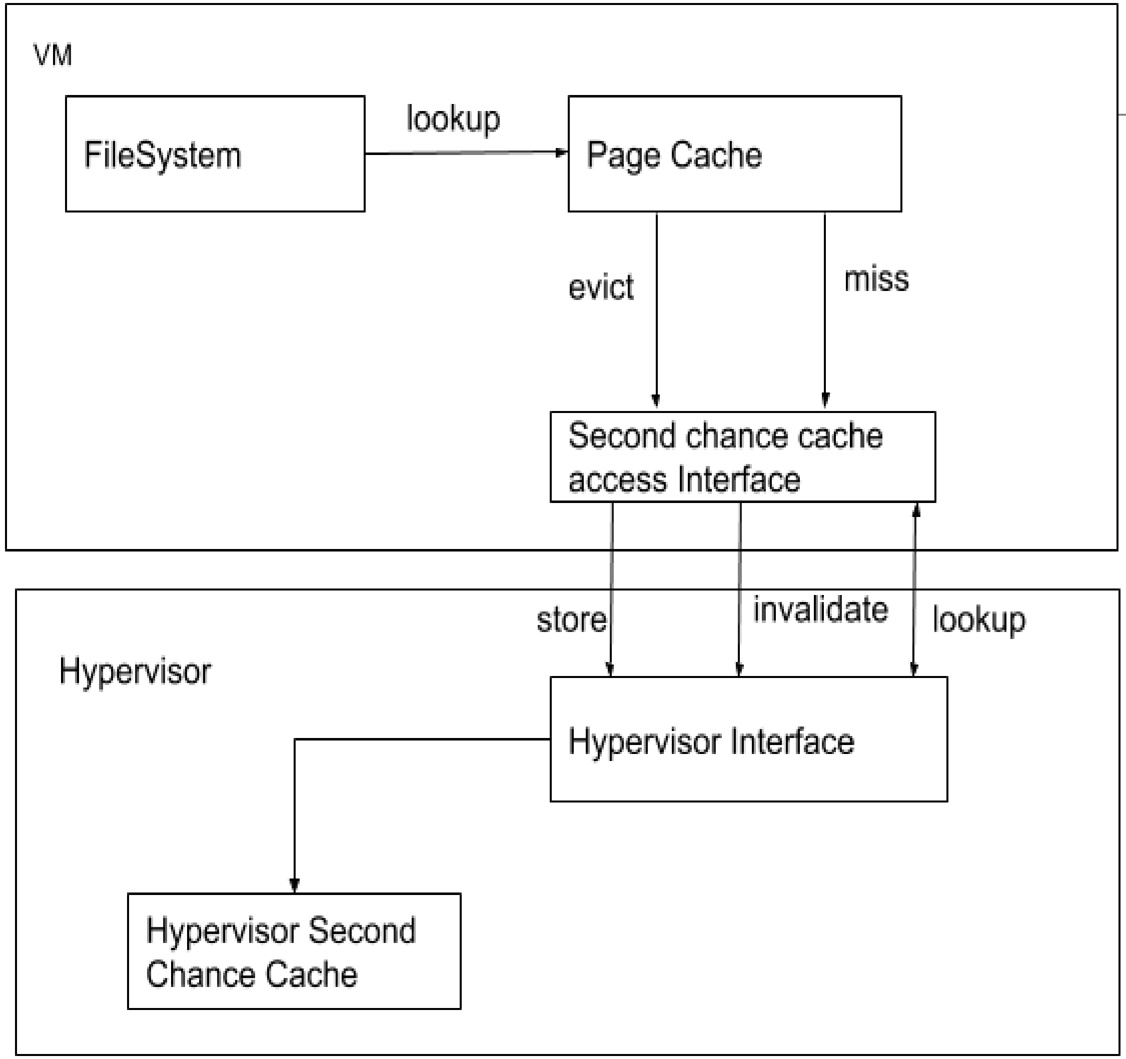


Fig 2:Second chance Cache. Adapted from DoubleDecker

- Exclusive Cache
- Can be used in single level as well as multi level control setups

Differentiated Cache in derivative cloud

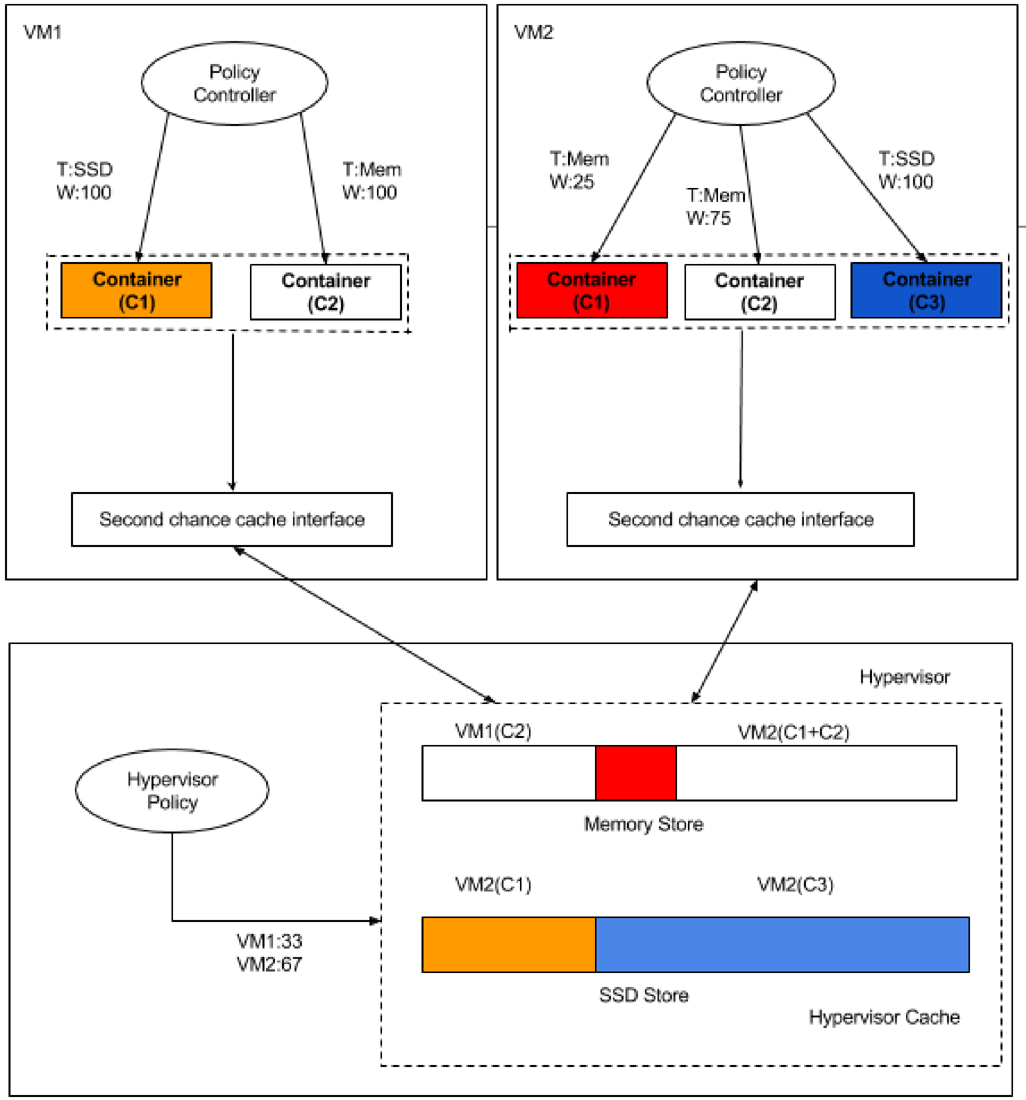
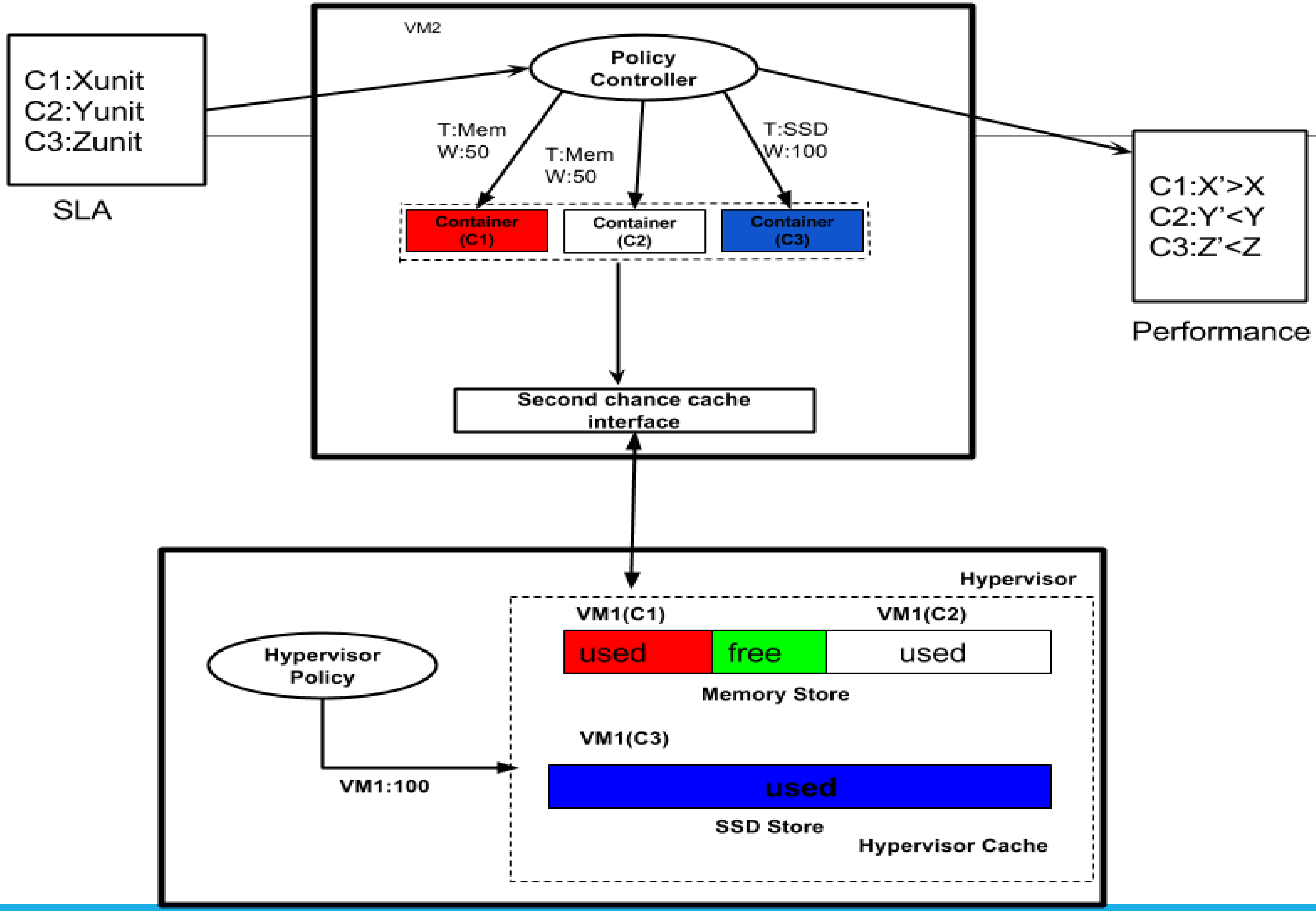


Fig 3: Differentiated cache framework for Derivative clouds. Adapted from DoubleDecker

- Partitioned cache at both the levels
- Satisfy the application objectives
- Memory as well as SSD cache back store

Challenges in Cache Provisioning



- Application behaviour not known
- SLAs not satisfied

Related Work

Hypervisor Caching

Comparative Analysis[3]	Hypervisor cache management capability in single level virtualized setup
Mortar[1]	Exposed hypervisor cache to applications by modifying them
Software Defined Caching[2]	Dynamic second chance cache provisioning satisfying application SLA in multi VM setup

Related Work

Derivative Cloud

Spot-Check[4]	Nested virtualization for minimizing the cost and availability tradeoff between spot servers and on-demand servers

Problem Statement

- Empirically knowing the application behaviour
- Provisioning cache statically to satisfy higher level application objectives

Experimental Evaluation of Application Characteristics

Hypothesis:

Applications behave same in isolation as well as simultaneous execution, if provided with same amount of resources

Experiment Design Requirements

Parameters:
- Cache size

Metrics of interest
- Application throughput
- Application latency
- Cache hit rate
- System resource utilisation

Workloads
- Webserver
- MongoDB
- Redis
- Mysql
- Webproxy

System Specification

- Host: Intel(R) 2.60GHz with 16 cores, 1TB HDD, 32GB RAM, Ubuntu 14.04 LTS(64 bit), Kernel 4.1.5, KVM hypervisor

- Guest: 13 cores of CPU, 128GB VDS, 16GB RAM, Ubuntu 14.04 LTS(64 bit), Kernel 4.1.5, lxc

Empirical Results Depicting Various Application Characteristics

Application Performance Without Cache

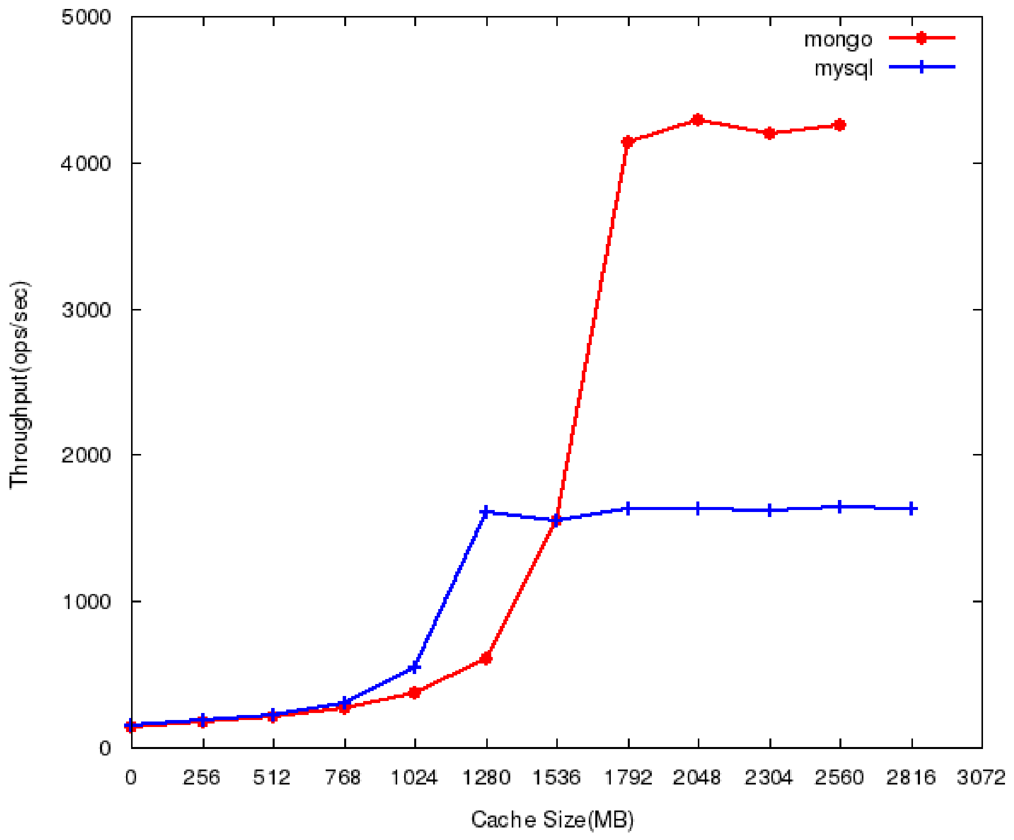
Application	Working Set Size(GB)	Throughput	Latency
Webserver	2.82	23.2 mb/s	42.8 ms
MongoDB	2.18	139.04 ops	7182.26 us
Mysql	1.37	155.00 ops	6438.01 us
Webproxy	2.50	23.8 mb/s	42.3 ms
Redis	-	90.47 ops	11044.35us

Memory Usage of Applications

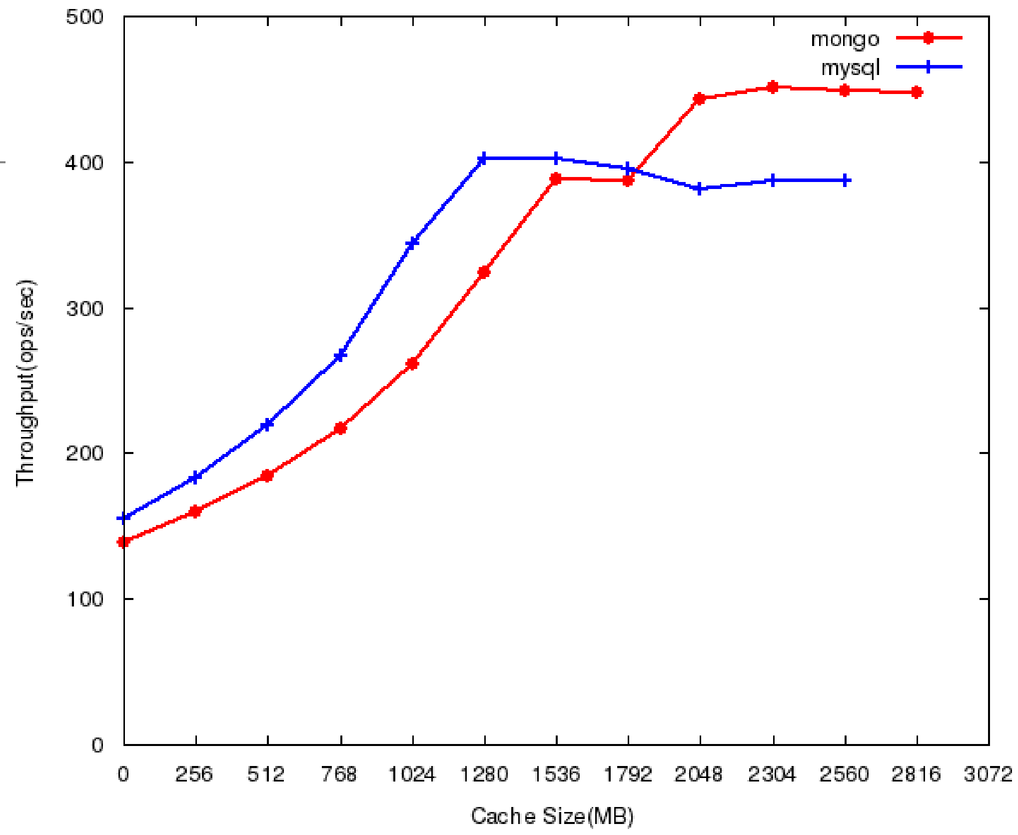
Application	Page Cache Memory(MB)	Anonymous Memory(MB)
Webserver	838.78	88.14
MongoDB	705.98	259.65
Mysql	387.00	566.61
Webproxy	765.00	77.34
Redis	1.57	952.10

Total Memory (MB)	953.67

Throughput vs Cache size

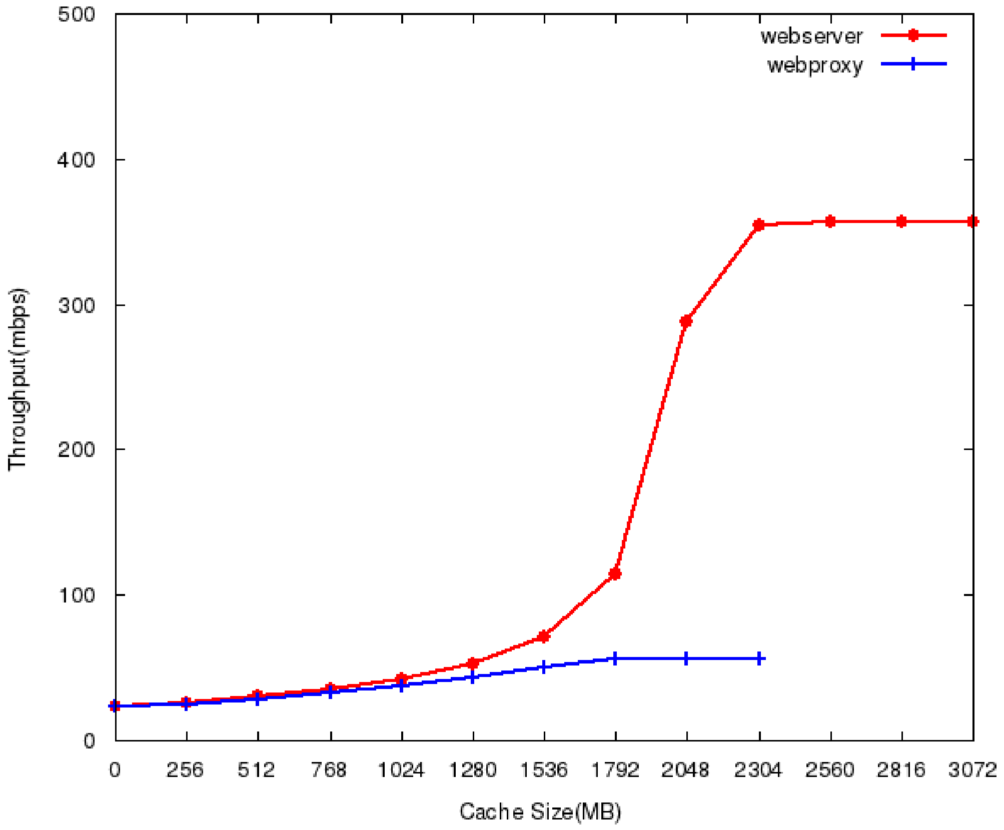


Memory Cache

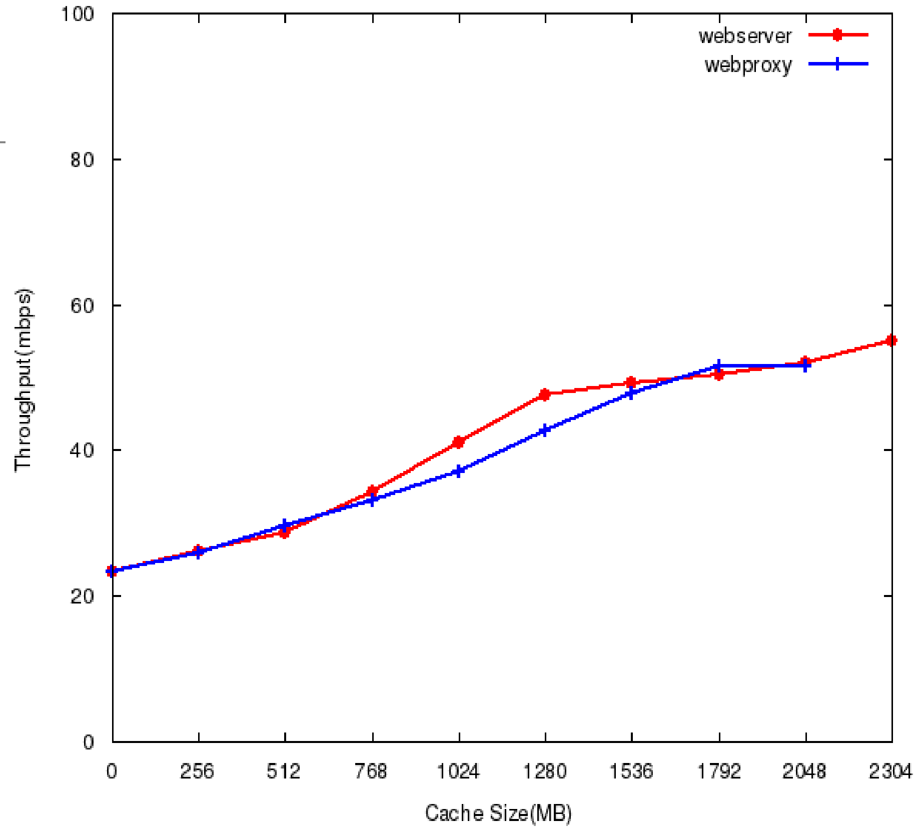


SSD Cache

Throughput vs Cache size

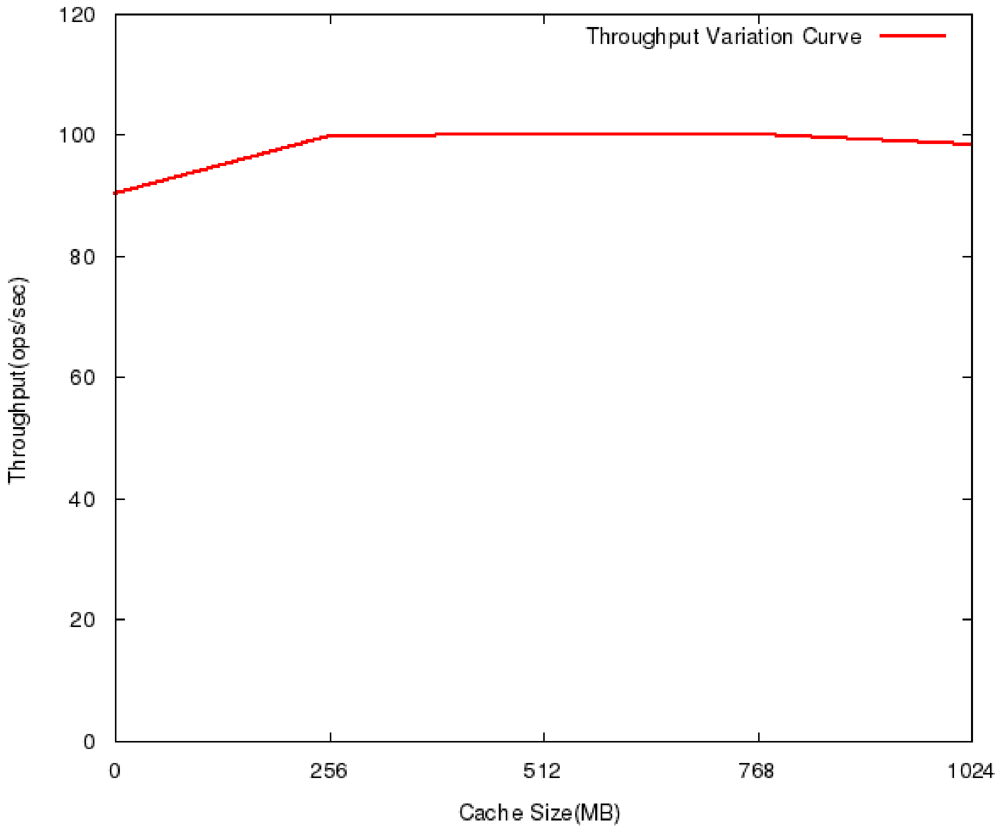


Memory Cache



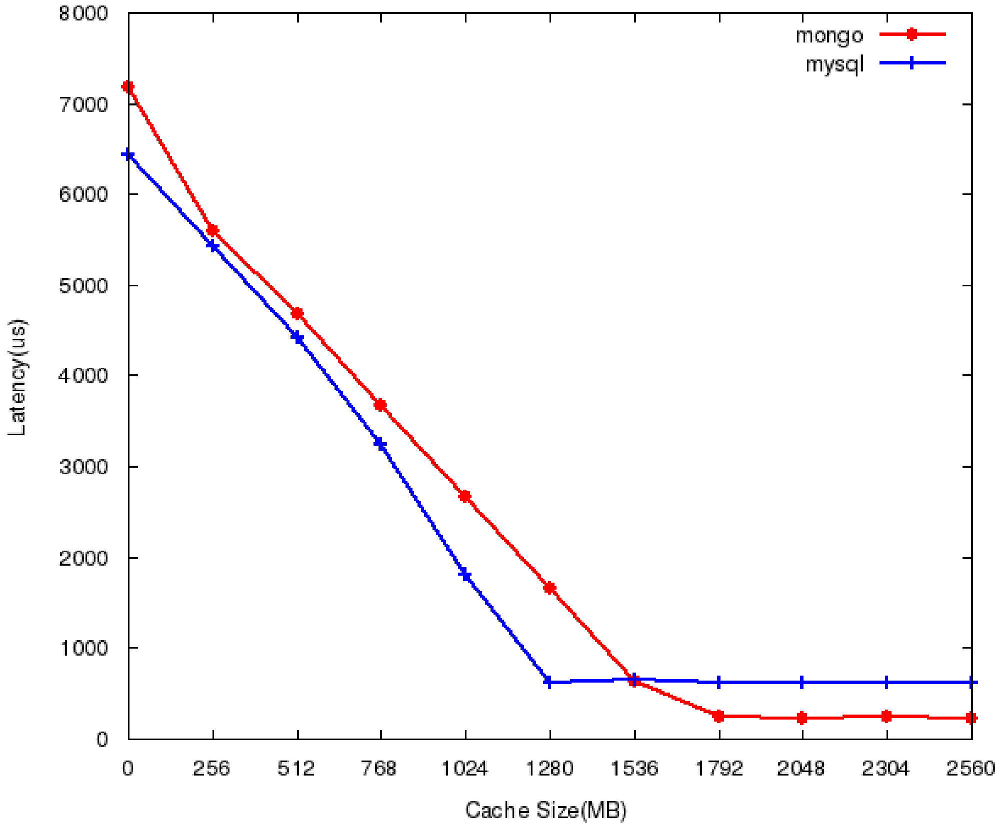
SSD Cache

Throughput vs Cache size Redis

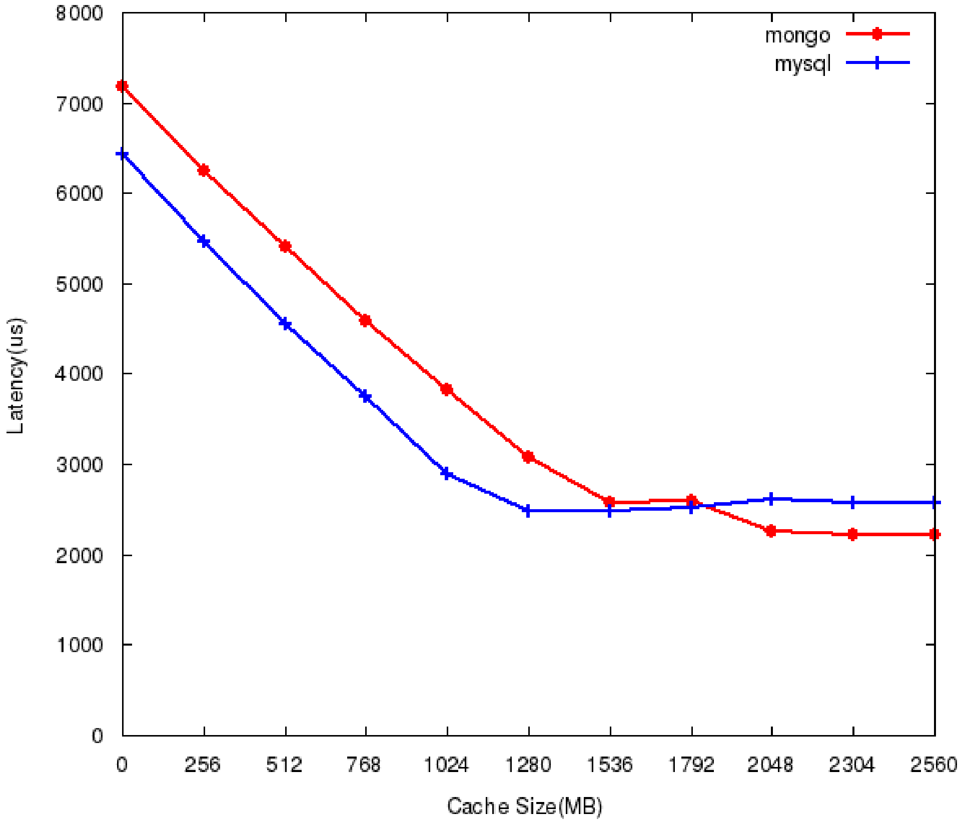


Memory Cache

Latency vs Cache size

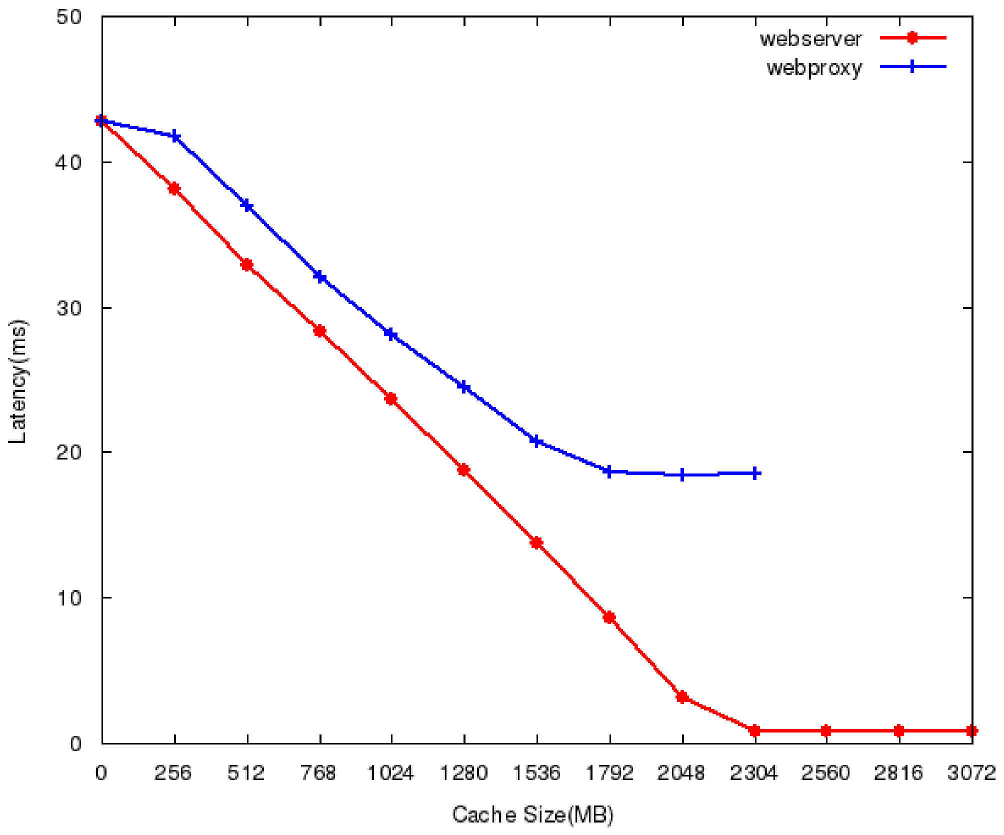


Memory Cache

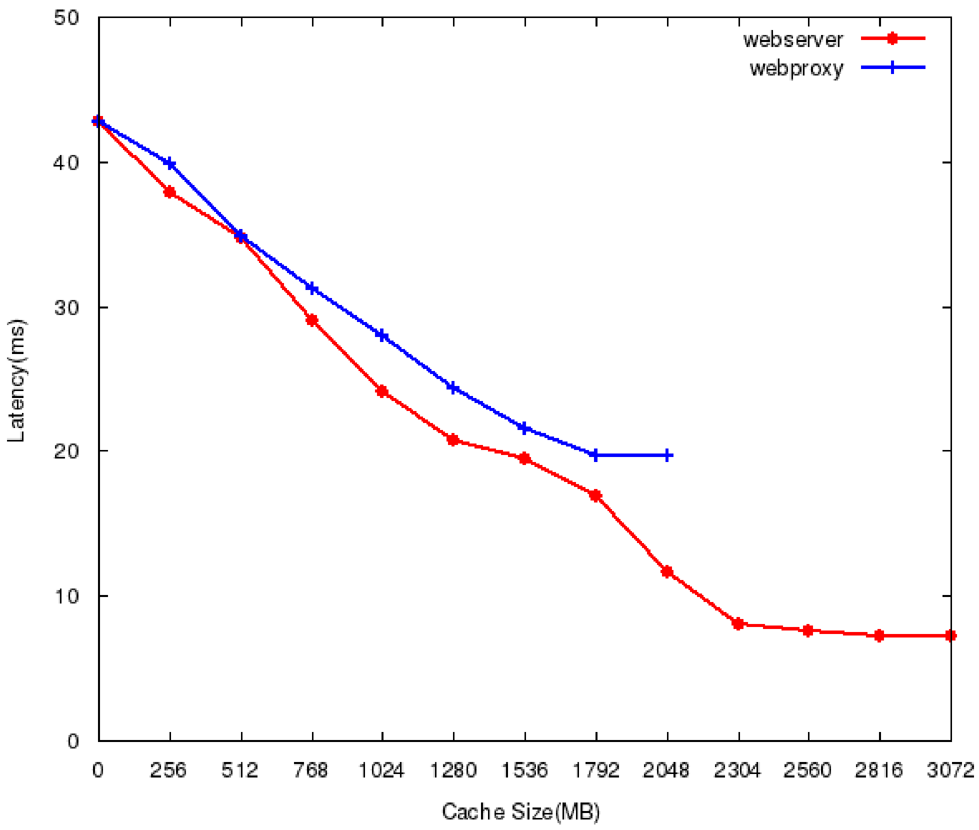


SSD Cache

Latency vs Cache size

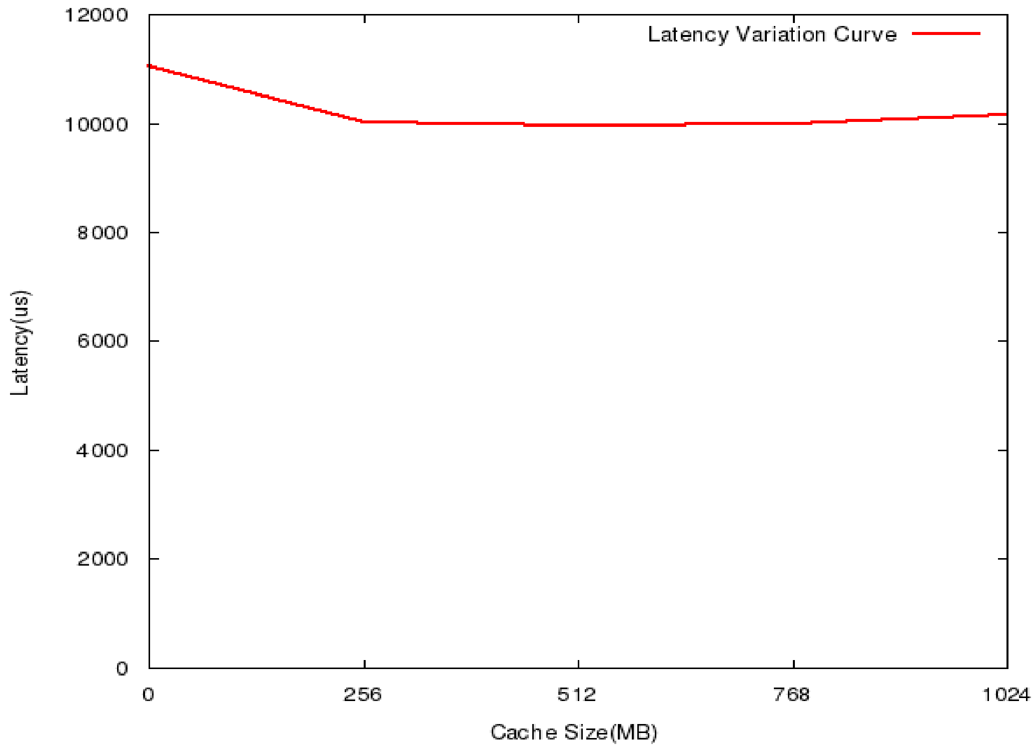


Memory Cache



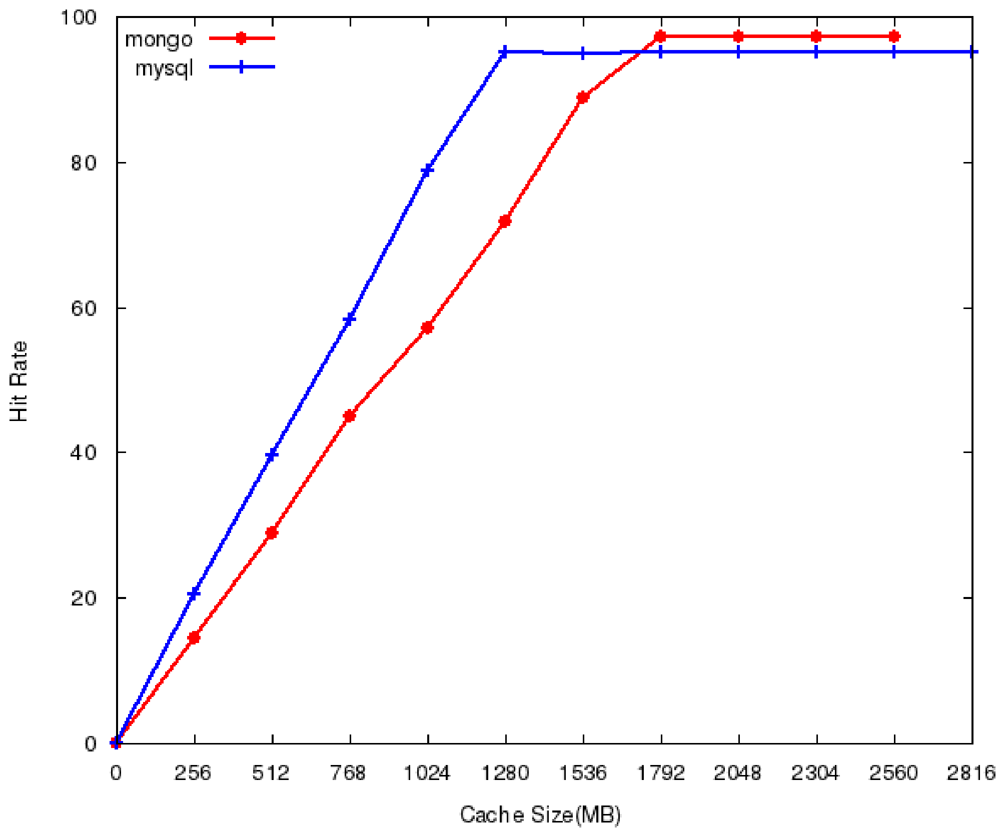
SSD Cache

Latency vs Cache size Redis

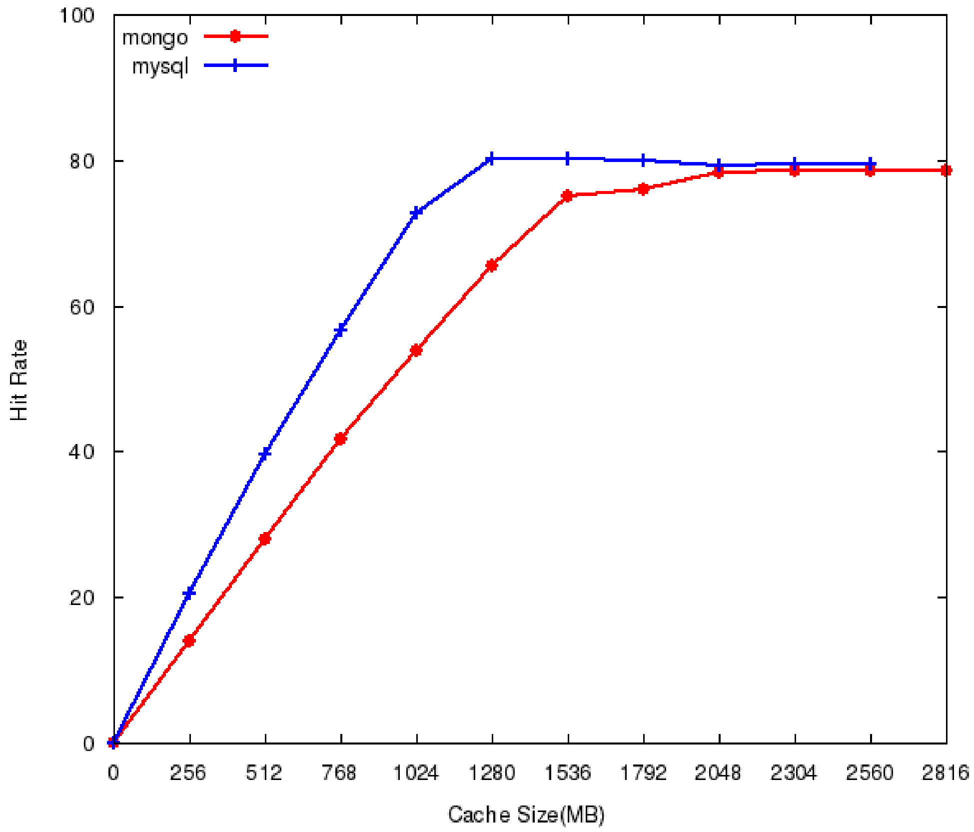


Memory Cache

Hit Rate vs Cache size

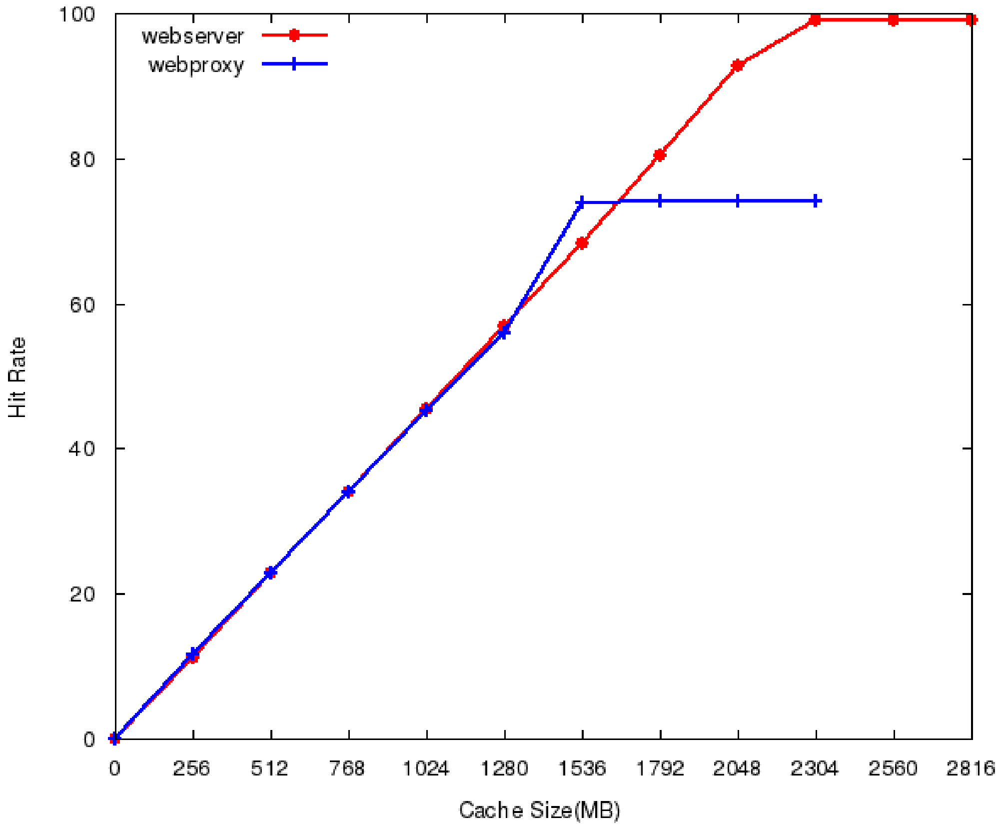


Memory Cache

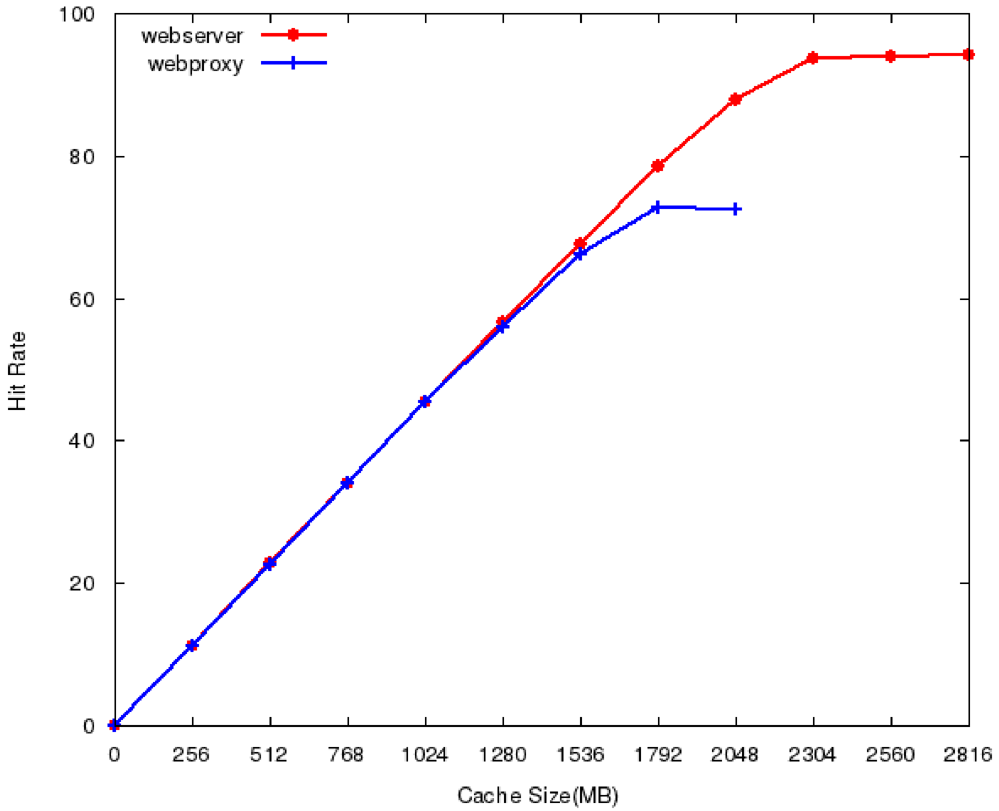


SSD Cache

Hit Rate vs Cache size

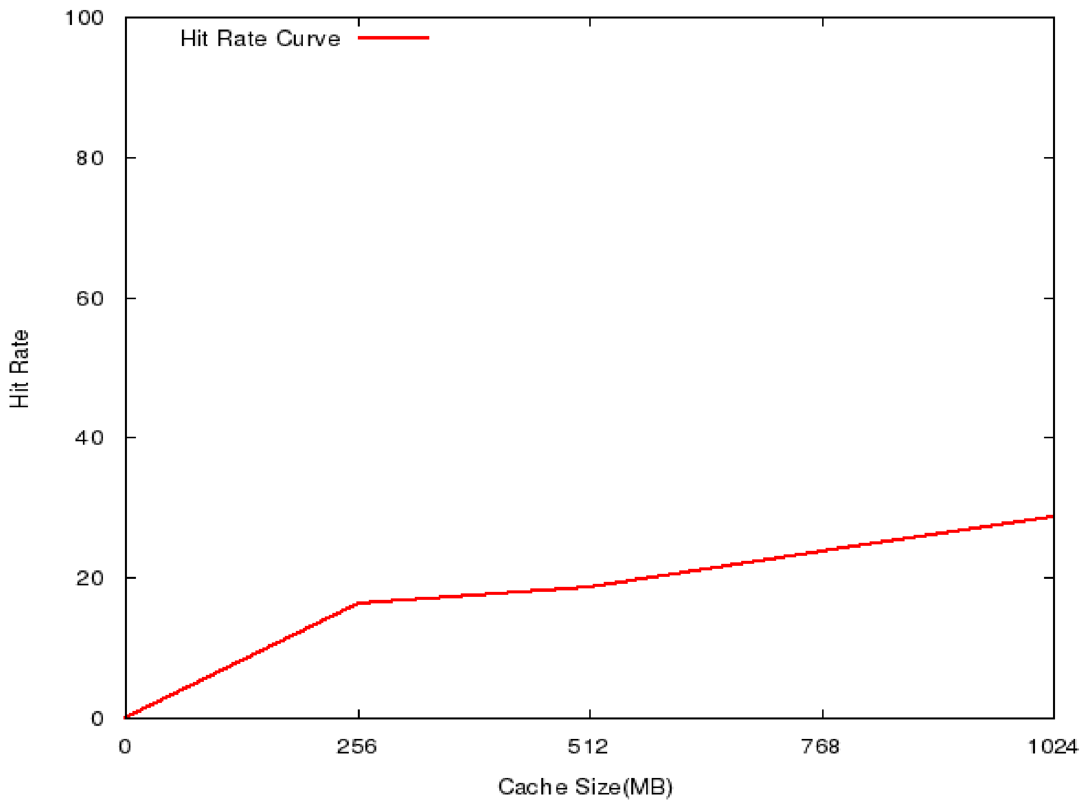


Memory Cache



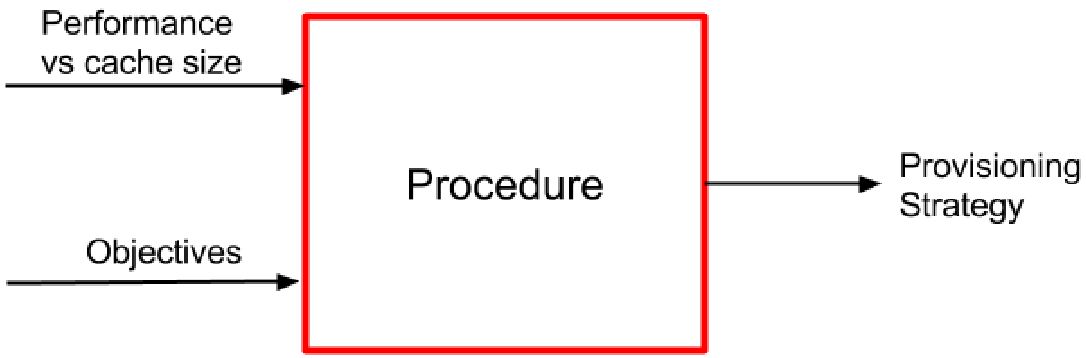
SSD Cache

Hit Rate vs Cache Size Redis



Memory Cache

Cache Provisioning Satisfying Application SLA



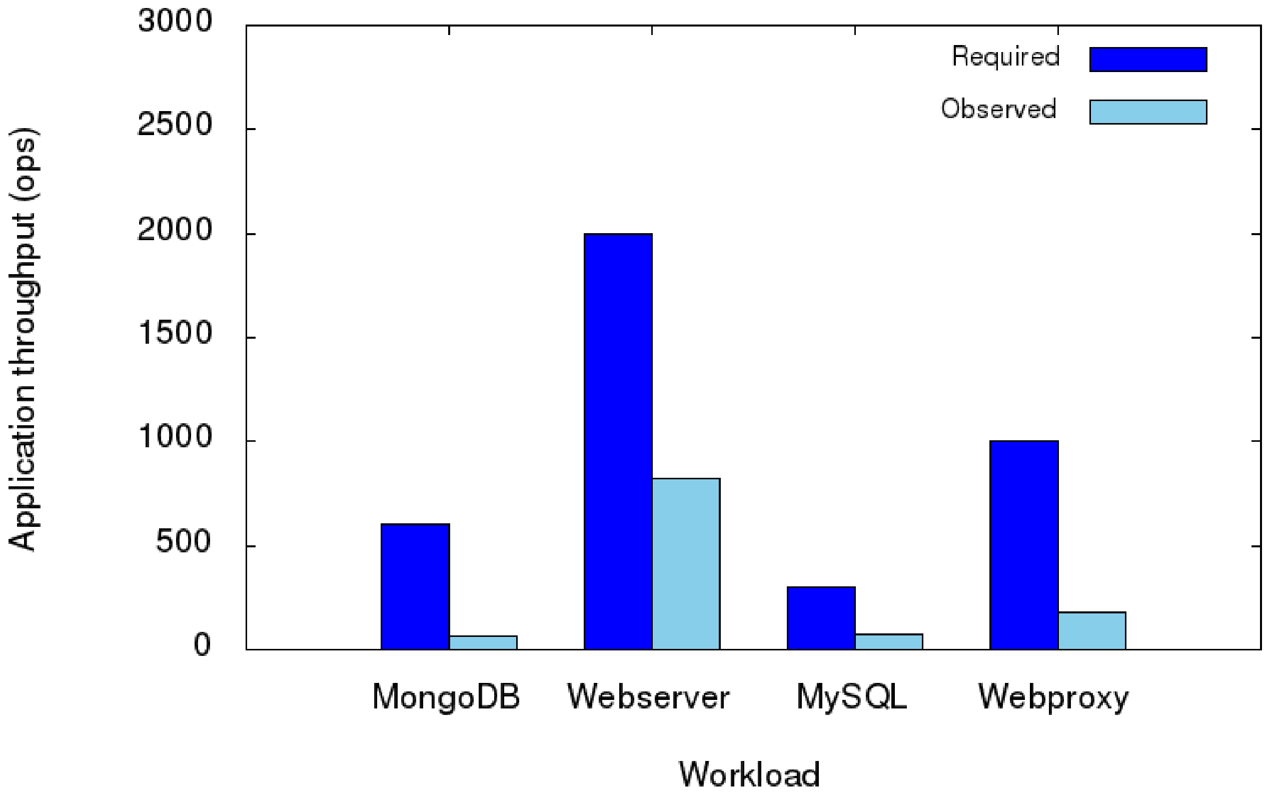
- Procedure is simple slope method to get the cache size for desired performance

First Set of SLA and Cache Allocation Required

Application	SLA(ops)	WSS(GB)	Memory Cache(MB)	SSD Cache(MB)
Webserver	2000	2.82	1605	0
MongoDB	600	2.17	1275	0
Mysql	300	1.37	743	1000
Webproxy	1000	2.5	1000	1227

Memory Cache	SSD Cache
4GB	1228MB

Experimental Results After Cache Allocation

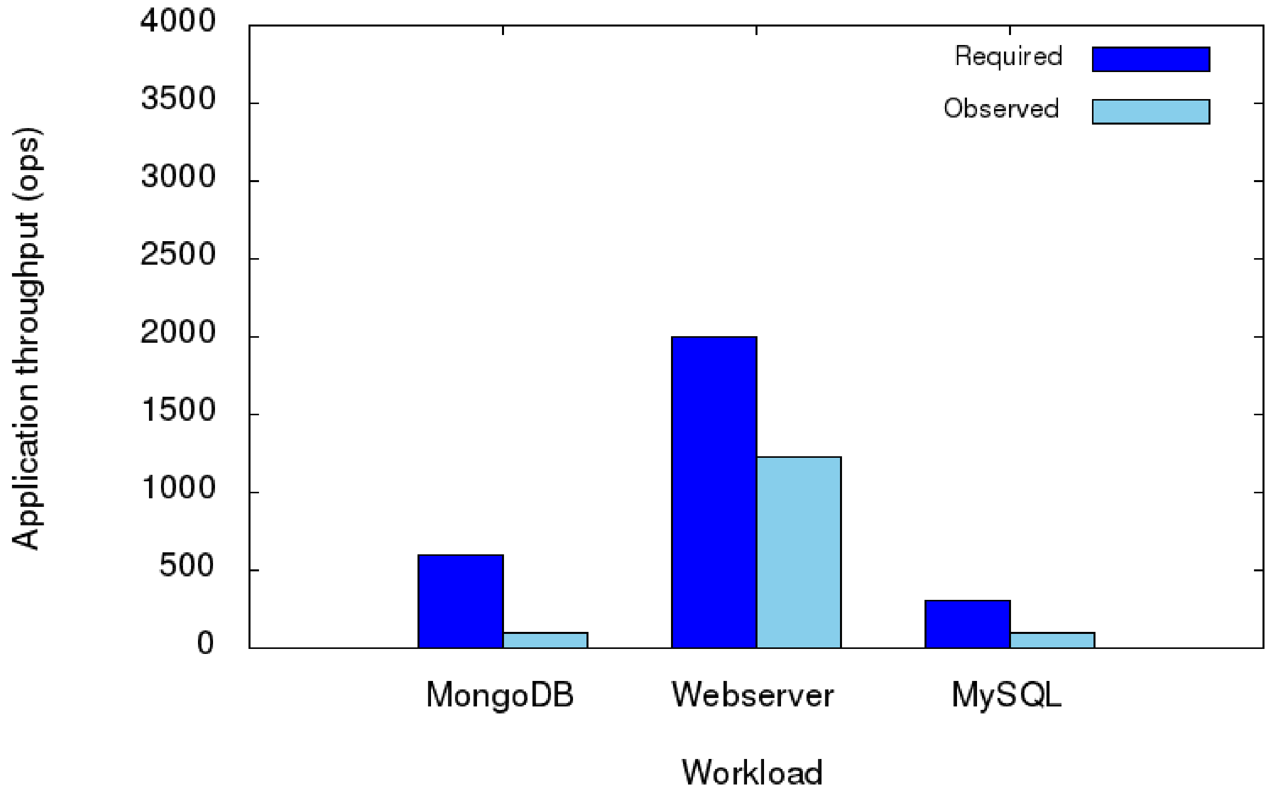


Second Set of SLA and Cache Allocation Required

Application	SLA(ops)	WSS(GB)	Memory Cache(MB)	SSD Cache(MB)
Webserver	2000	2.82	1605	0
MongoDB	600	2.17	1275	0
Mysql	300	1.37	743	1000

Memory Cache	SSD Cache
3GB	1000MB

Experimental Results After Cache Allocation

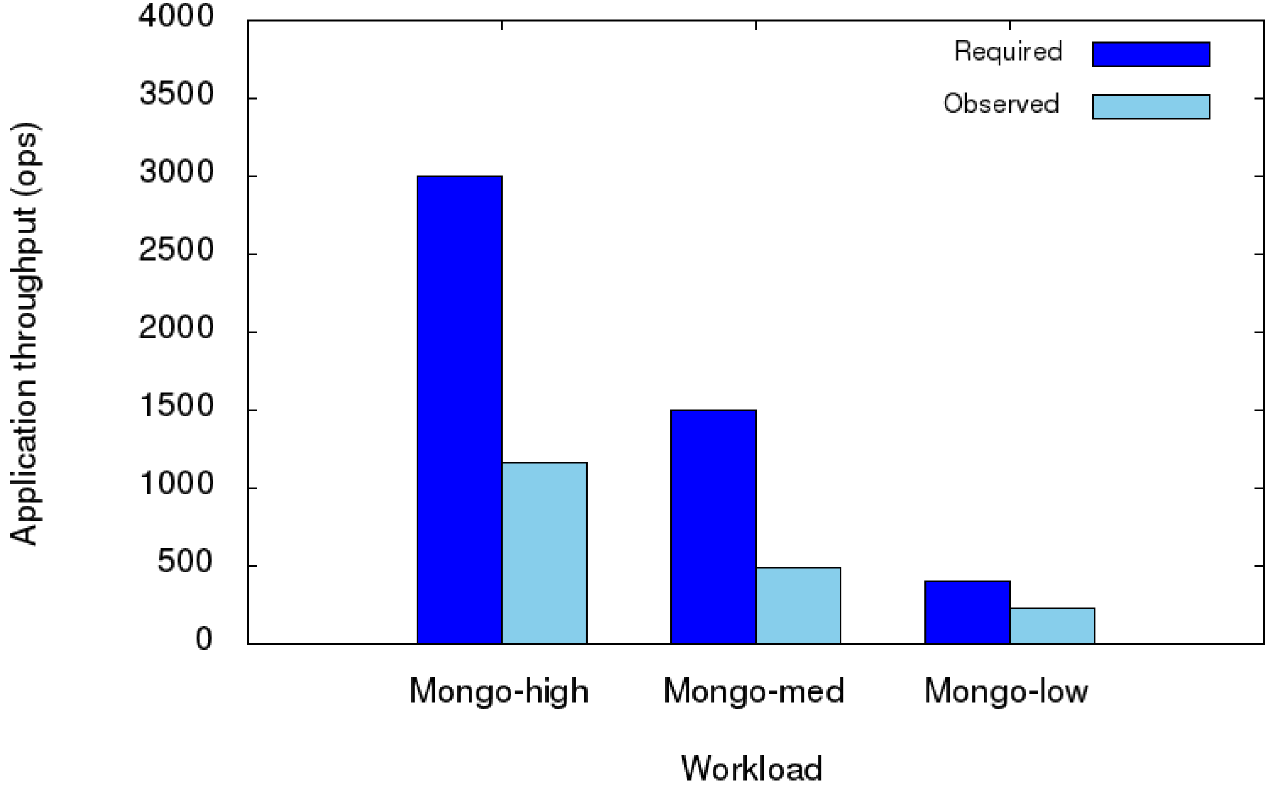


Third Set of SLA and Cache Allocation Required

Application	SLA(ops)	WSS(GB)	Memory Cache(MB)	SSD Cache(MB)
Mongo_high	3000	2.17	1680	0
Mongo_medium	1500	2.17	1522	0
Mongo_low	400	2.17	1053	1821

Memory Cache	SSD Cache
4GB	1821MB

Experimental Results After Cache Allocation

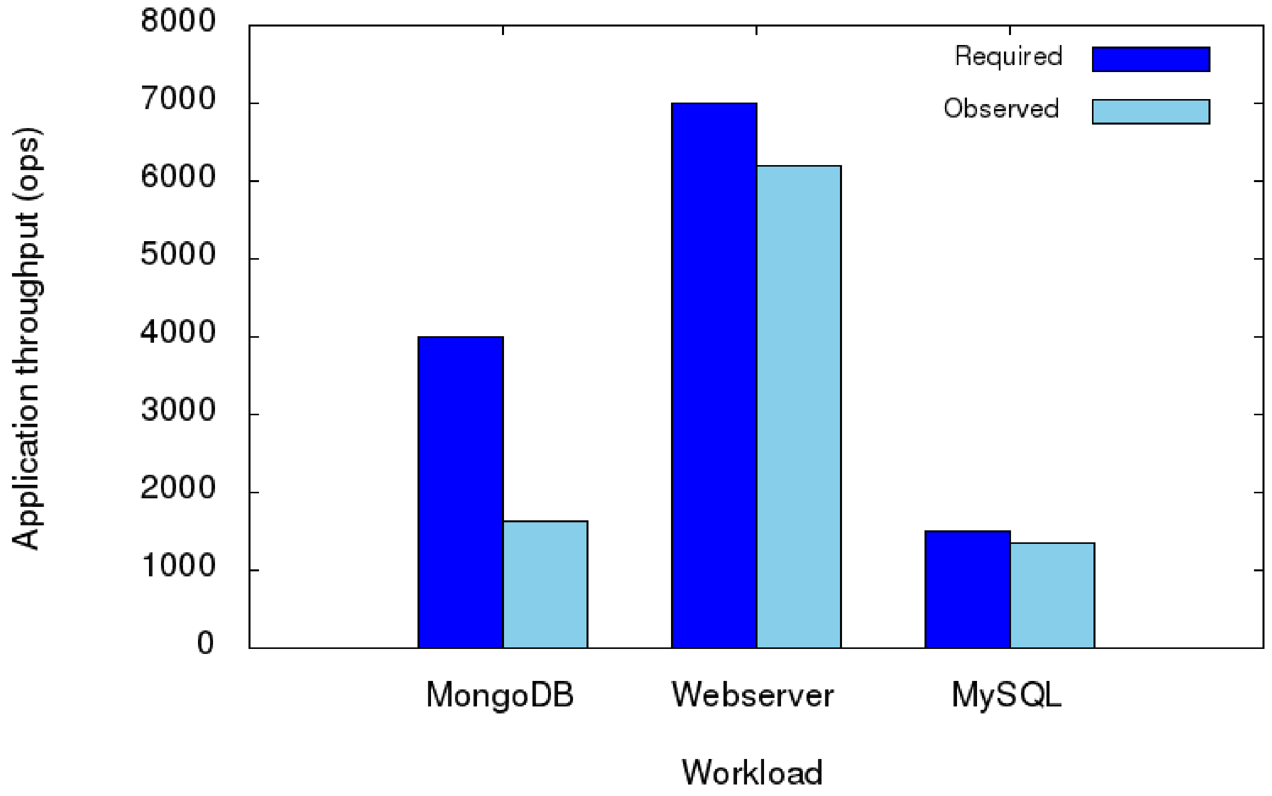


Fourth Set of SLA and Cache Allocation Required

Application	SLA(ops)	WSS(GB)	Memory Cache(MB)	SSD Cache(MB)
Webserver	7000	2.82	2060	0
MongoDB	4000	2.17	1779	0
Mysql	1500	1.37	1253	0

Memory Cache	SSD Cache
5GB	0

Experimental Results After Cache Allocation



Hypothesis Failure

- I/O wait of CPU for most time of the run phase
- For fourth setup core allocated to mongoDB was idle due to I/O wait for 50% of time.

Conclusion

- Empirical study of differentiated cache provisioning framework
- Behavioural analysis of different applications
- Statically provisioning cache,but due to failed hypothesis, experiment redesigning is required which takes into account disk contention

Future Work

- Redesigning experiments with correct hypothesis
- Giving an algorithm for satisfying individual as well as having a global maxima for performance
- Dynamic cache provisioning

References

1. Hwang, J., Uppal, A., Wood, T. and Huang, H., 2014. Mortar: Filling the gaps in data center memory. *ACM SIGPLAN Notices*, *49*(7), pp.53-64.
2. Stefanovici, I., Thereska, E., O'Shea, G., Schroeder, B., Ballani, H., Karagiannis, T., Rowstron, A. and Talpey, T., 2015, August. Software-defined caching: Managing caches in multi-tenant data centers. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (pp. 174-181). ACM.
3. Mishra, Debadatta, and Purushottam Kulkarni. "Comparative analysis of page cache provisioning in virtualized environments." *Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2014 IEEE 22nd International Symposium on*. IEEE, 2014.
4. Sharma, Prateek, et al. "Spotcheck: Designing a derivative iaas cloud on the spot market." *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015.

Thank You