WHITEPAPER

MEMSTAC – A BRIEF PRIMER

TABLE OF CONTENTS

MEMSTAC™

Software Stack for High Performance, Big Data Infrastructure Caching

Flash-based Caching Solution for In-Memory applications, delivering:

- *Higher cache performance across varying workloads*
- *Reduced memory cost per server*
- *Lower power consumption*
- *Higher cache capacity per node*
- *DRAM-like throughput and latency for 10 GbE across various K-V sizes*
- *Reduced total-cost-of-ownership over 3 years by over 70%*

OVERVIEW

Modern cloud datacenters use extensive low-latency access DRAM caching to improve application performance. Often, insufficient cache capacity leads to excessive networking and database querying overheads, thereby slowing application performance. While DRAM-only solutions deliver excellent response time, they are relatively expensive, high power, and low capacity. By contrast, NVMe SSDs are relatively inexpensive, low power, and high capacity, but significantly lower performance than DRAM in native use.

OmniTier offers a function-optimized, cluster-based software stack, MemStac, that enables the use of NVMe SSDs, with a small fraction of the data resident in DRAM, for in-memory caching in the cloud. MemStac is capable of exceeding 7 million cache requests per second and supporting up to 4 terabyte cache capacity per server node at a fraction of the cost of today's DRAM-only solutions. MemStac is fully Memcached compliant and can be integrated into existing cache clusters seamlessly, requiring no changes to the existing application (client) libraries.

Such tiered-memory solution enables datacenter operators to optimize for both capacity and response times to deliver higher application performance at a fraction of the cost of today's solutions. MemStac works with standard Intel processors and select off-the-shelf NVMe SSDs.

MemStac is designed to support two Open Source in-memory caching protocols, including:

1. **Memcached** - a widely deployed memory object caching system. MemStac is protocol compliant with standard Memcached, enabling it to work seamlessly with existing Memcached environments. It supports cluster configurations for scaleout based on node-to-node or proxy node connectivity. This protocol was generally available at year-end 2016.
2. **Redis** – a growing Open Source in-memory key-value store that supports data structures such as sorted sets and lists. MemStac supports prescribed fault tolerance and self-healing using Master/Slave replication across zones or regions. Redis capabilities will be added to the MemStac family of protocols during 2017.

Caching improves application performance by storing critical pieces of data in the form Key-Value (K-V) in memory for low-latency access. Cached information may include the results of I/O-intensive database queries or the results of computationally-intensive calculations.

MemStac is designed to emulate the performance of Open Source stacks that support DRAM-based architecture.

IN-MEMORY CACHE PERFORMANCE

BIGGER CACHES DELIVER CONSISTENTLY BETTER PERFORMANCE ACROSS VARYING WORKLOADS

The concept of in-memory caching is shown in Figure 1 below:

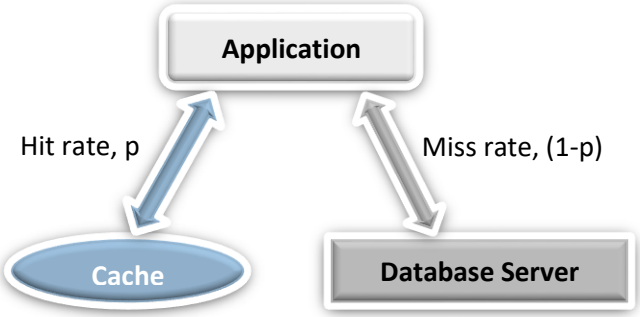

Figure 1 In Memory Cache Concept

Active data for an application is brought into high-speed cache servers for faster access. The application processing is then carried out by retrieving the needed data either from the cache server or the database server. Ideally, all data required to service an application should reside in the cache cluster. However, such a proposition could be prohibitively expensive. As such, due to capacity limitations of the cache, not all data is available from the cache servers; some missing data is retrieved from the slower database server.

If p is the probability of the application data being available on the cache servers, then (1-p) is the probability of the data that is retrieved from the database server. The miss rate, (1-p), is a measure of the overhead due to networking and database queries. It can be minimized by maximizing p, which again points to the idea of provisioning larger caches for accelerating application performance.

The average response time per request seen by the application is given by:

$$T_{avg} = pT_{cache} + (1 - p)T_{DB} \qquad (1)$$

where T_{cache} and T_{DB} are the average response times of the cache and the database, respectively. When p=0 (that is, no caching), the response time seen by the application equals the average response time of the database. On the other hand, when all active data is in the cache, with p=1, the average application response time equals the average cache response time.

These two response times are dictated by the type of media used with the respective servers. In order to achieve small T_{cache}, the cache servers today are typically provisioned with massive amounts of expensive and power-hungry DRAM, which make the provisioning of large cache capacity an expensive proposition. On the other hand, database servers typically use the lower cost hard disk drives to store the data and, as such, exhibit much larger average response times

than T_{cache}. Indeed, T_{DB} often depends upon the request rate (load) to the database server, and can exhibit a thresholding effect wherein it increases rapidly beyond a prescribed request rate. Such regime of operation is to be clearly avoided, requiring sufficient cache capacity be provisioned to ensure that the request rate directed to the database servers is kept below the threshold level.

How much cache capacity is sufficient? The answer depends upon the desired value of p and the workload characteristics. Figure 2 below shows the dependence of p for different static workloads, modeled by a Zipf-like probability density function:

Zipf-like Workload Model: $\Pr(jth\ object) = C/j^{\alpha}$

where C is a normalizing constant and 0< α <=1 defines the K-V frequency distribution.

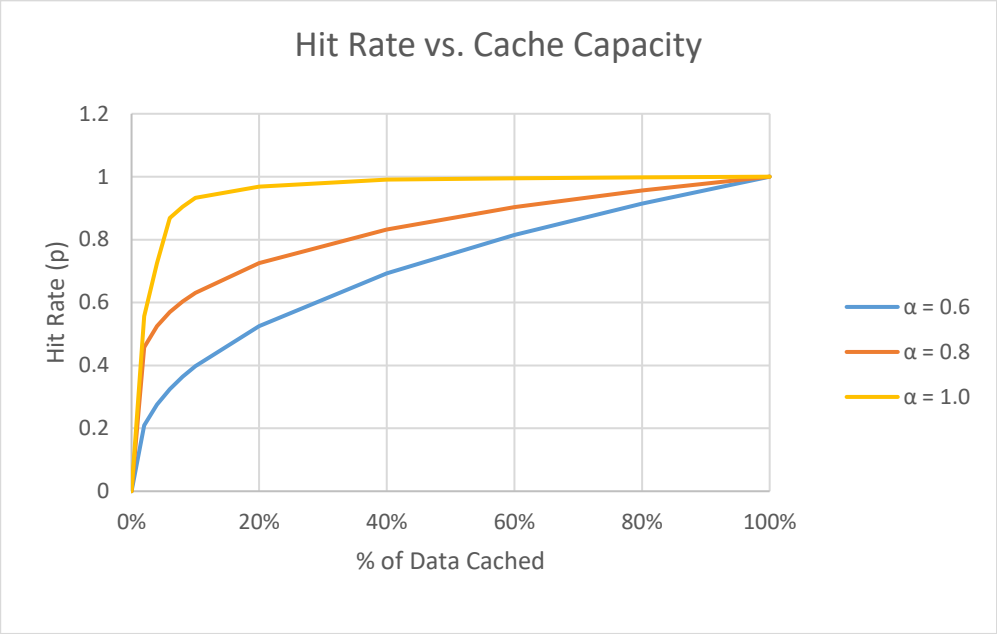

Figure 2 Hit rate dependency on cache size and workload distribution

Note that the hit rate not only depends upon the cache capacity (the x-axis, representing the fraction of the database cached), but also on the workload type, as characterized by the parameter α. If the data is highly structured (e.g., α = 1), caching of only 10% of the most probable data objects will suffice to yield a 90% hit rate. However, if the workload changes to α = 0.6, the hit rate would drop to 40% for a cache capacity of 10%, thereby possibly overloading the database server and slowing the application performance dramatically.

Figure 3 below shows the associated behavior of the average response at the application level as a function of the percentage of data cached for different workloads. The database average

response time is assumed to be 15x of the corresponding cache response time (normalized to 1 unit) in the figure.

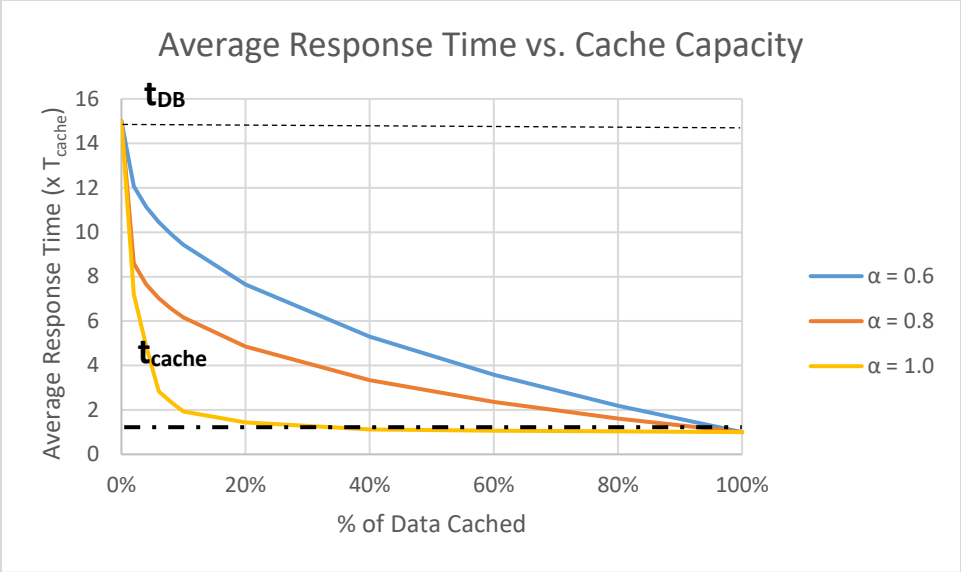

Figure 3 Application's average response time dependency on cache size and workload

As expected, highly structured data (α = 1) achieves near cache response time with roughly 20% of data cached. However, for unstructured workloads, the cache size needs to be between 60% and 80% of the active data objects to achieve performance similar to that for more structured workloads.

As more diverse, data-intensive applications demanding consistently high performance get serviced in modern datacenters, the need for larger cache capacities will inevitably become a "must have". Such requirement can become prohibitively expensive with today's DRAM-only architecture. Not only is DRAM expensive, it is also power hungry; and, hence, limited in terms of capacity per server that can be provisioned in in-memory clusters due to power consumption and space considerations.

By enabling the use of a tiered memory architecture using NVMe SSDs with a small amount of DRAM, MemStac overcomes the drawbacks of the DRAM-based solutions and offers the following benefits:

- Reduced memory cost per server
- Lower power consumption
- Higher cache capacity per node
- Reduced acquisition and total-cost-of ownership for a prescribed request rate
- Higher and consistent application performance across varying workloads

PERFORMANCE SUMMARY

MEMSTAC SATURATES 10GBE REQUEST RATE OVER PRACTICAL RANGE OF KEY-VALUE SIZES

Figure 4 below summarizes the Read performance of MemStac relative to the bounds on the requests per second placed by the 10 GbE networking environment.

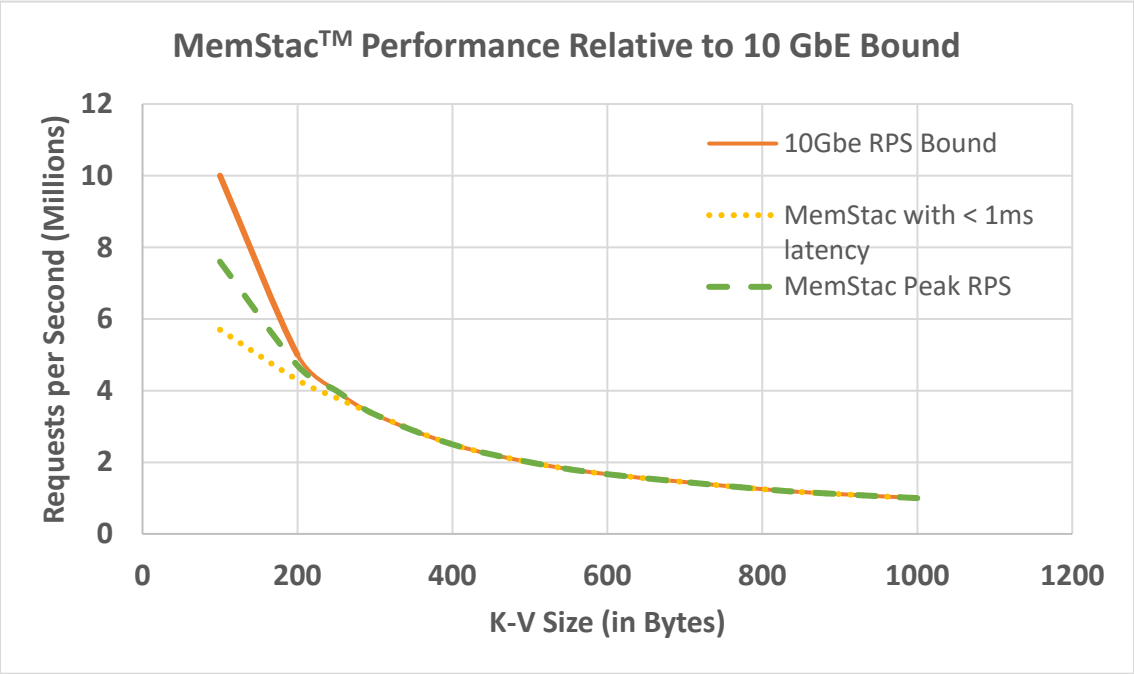

Figure 4 MemStac Read Performance vs. K-V Size (100% Read)

As shown, the throughput delivered by the network decreases monotonically with increasing length of the Key-Value (K-V). Using select off-the-shelf NVMe SSDs as cache memory within a standard Intel dual-socket server, MemStac is able to saturate the combined IOP capability of the stacked SSDs. As an example, with prescribed off-the-shelf NVMe SSDs, MemStac achieves 3.7M requests per second (100% reads) and 3.2M requests per second (80% reads / 20% writes) for 100 byte records. With commonly-used workloads and employing OmniTier's proprietary data classification algorithms, the performance increases to an industry-leading 6.1M and 4.7M operations per second, respectively, using DRAM storage at one-eighth of the cache capacity. These levels of performance are achieved with less than 1 millisecond average latency.

Figure 4 summarizes the Read performance of MemStac across different K-V sizes. As shown, MemStac seamlessly delivers 10 GbE network-limited throughput with typical workloads exceeding 250 bytes in average size at average latency of 1ms or less. Below 250B, the achievable throughput diverges from the network bound, but still remains close to Open Source DRAM-only solutions.

FEATURES

DROP-IN REPLACEMENT OR ADDITION

MemStac is Memcached or Redis compliant. As such, it can be a) deployed as a drop-in replacement for existing Memcached or Redis environment, or b) added to an existing Memcached or Redis cluster to support scaleout.

COST EFFECTIVE

By largely using NVMe SSDs for caching, MemStac enables the use of lower cost servers to achieve a prescribed level of throughput.

MAINTENANCE AND DIAGNOSTICS

MemStac provides detailed monitoring statistics for the nodes in the cluster to support diagnostics and maintenance.

ELASTIC

MemStac supports console-based management of the cluster, such as addition or deletion of cache nodes to meet the prescribed performance and capacity requirements.

SNAPSHOTS

MemStac creates snapshots of the cluster to provide data protection and re-use for cache initialization, etc..

RELIABLE

MemStac offers features that enhance reliability for mission critical deployments, including automatic failure detection and recovery.

CLUSTER SUPPORT

MemStac supports two types of cluster configurations for Memcached deployments: 1) Client library-assisted cluster, which is based on standard Memcached library sharding, and 2) Proxy-assisted cluster, which is based on the use of intermediate proxy nodes (e.g., McRouter). For Redis, MemStac supports prescribed fault tolerance, replication, and self-healing features.

LOAD BALANCING

MemStac uses sophisticated data classification and monitoring algorithms to provide load balancing across the cluster.

USE CASES

MEMSTAC'S TIERED MEMORY SOLUTION MAKES BIGGER CACHES AFFORDABLE, ENABLING HIGHER APPLICATION PERFORMANCE

MemStac is designed to enable dramatic performance improvement or cost savings, or both, for the in-memory cache tier in the cloud. Using cost estimates from Uptime Institute (True TCO Model from uptimeinstitute.com) for server, memory, and annual Opex listed in Table 1, we estimate that the MemStac-based Memcached solution delivers approximately 66% and 70% savings in acquisition cost and TCO over 3 years, respectively, relative to the Open-Source DRAM-based solution in relatively high performance, large capacity infrastructure deployments.

Material	Estimated Cost ($)	Material	Estimated Cost ($)
DRAM + SW	10 per GB	Server	2000
NVMe SSD + SW	2 per GB	Annual Opex per server	3422

These dramatic savings offer infrastructure providers the opportunity to benefit in two ways:

1. ***Introduce a higher tier of Service Level Agreement (SLA) with associated higher service fees:*** The NVMe SSD-based architecture enables reduced cost, power, and space per server. This allows the infrastructure service provide to provision larger caches for a prescribed capital outlay. As illustrated in Figure 4, larger caches deliver improved and consistent response times and associated SLAs across varying workloads.
2. ***Reduce the infrastructure costs for supporting existing SLAs:*** MemStac is designed to emulate the performance of Open Source stacks that support DRAM-based architecture. As such, by using MemStac with SSD-based servers, the infrastructure service provider can reduce cost, power, and space without sacrificing the performance.

In both cases, MemStac can contribute to the bottom-line growth of cloud infrastructure providers.

Since MemStac is designed to be compliant with Open Source Memcached and Redis, it can be used to significantly improve latency and throughput for many read-heavy or compute-intensive applications, such as social networking, gaming, media sharing, Q&A portals, IoT, autonomous cars, medicine, and the like. Some illustrative applications include:

DISTRIBUTED CACHE FOR WEB APPLICATIONS

When a webserver receives a request, it needs to send multiple database queries, in the form of Key-Value pairs, to create the page for the visitor. This process is computationally intensive for database servers to handle all of the requests directly. As such, webservers use caching to improve application performance by storing frequently accessed pieces of data in the in-memory cache cluster instead of repeatedly going to primary data store. Popular frameworks like Django,

Magneto, WordPress support the Memcached protocol, and MemStac can be used directly with these applications.

CONTENT DELIVERY NETWORK

Content Proxy servers act as a gateway between the user and origin/source servers to cache documents such as files, images, and html pages. When a user attempts to access the content from the origin server, the proxy server checks its internal cache for a recent copy of the requested Key-Value pair. If the content is available, it is delivered to user directly. If the content is not available, it is retrieved from the origin servers and simultaneously cached into proxy cache servers. MemStac supports wide range of Key-Value sizes, and can be deployed with proxy servers to provide higher performance and improved Service Level Agreements (SLA).

HIGHLY AVAILABLE CACHE CLUSTER

MemStac supports highly available, scale-out cluster deployments. The cluster can be dynamically scaled to meet application demands and also provide automatic fault recovery on node failures. A MemStac SERVER CLUSTER can be used as large temporal cache that enables processing of very large datasets in diverse applications in medicine, automotive, transportation, smart factories, and other areas.

ABOUT OMNITIER INC.

OmniTier Inc., founded in 2015, is a developer of high-performance, function-optimized solutions for modern datacenter infrastructure applications, including data caching, NoSQL, and real-time analytics using novel memory-management architectures. Its leadership team has a track record of delivering many “industry firsts” in data storage and access across different media types. The company currently has offices in Santa Clara, California, and Rochester, Minnesota.