SOLUTION BRIEF

BUILDING A HIGHLY EFFICIENT REDIS ON FLASH CLUSTER WITH SUPERMICRO X12 FATTWIN® SERVERS

Supermicro X12 FatTwin Server Platform

TABLE OF CONTENTS

Executive Summary / Introduction to Redis On Flash 1

The Influence of Flash on Cluster Design 2

CSD 2000 in the U.2 Form Factor 3

Redis on Flash Hardware Configuration and Results 4

Conclusion 17

SUPERMICRO

As a global leader in high performance, high efficiency server technology and innovation, we develop and provide end-to-end green computing solutions to the data center, cloud computing, enterprise IT, big data, HPC, and embedded markets. Our Building Block Solutions® approach allows us to provide a broad range of SKUs, and enables us to build and deliver application-optimized solutions based upon your requirements.

Executive Summary

This solution brief discusses Redis on Flash (Scaleflux computational storage) on the Supermicro X12 FatTwin Server and the performance and cost benefits that this configuration provides to data center customers.

Introduction to Redis on Flash

Redis on Flash increases the capacity of a Redis cluster by utilizing high performance Flash memory as a storage tier below main memory (RAM). All keys are stored in RAM along with the hottest values. Cooler values are migrated to Flash. When a value stored in Flash is accessed, it is promoted back into RAM. More details about the internal operation of Redis-of-Flash can be found at https://redislabs.com/blog/hood-redis-enterprise-flash-database-architecture.

The Supermicro X12 FatTwin® architecture provides flexibility and system accessibility for unique data center requirements. Unique one-half width nodes provide two nodes per rack unit, allowing for maximum reliability for modularized left and right nodes with redundant power supplies. These highly modular multi-node systems feature a tool-less design, and each node supports dual 3rd Gen Intel Xeon Scalable processors for improved performance. Redis on Flash utilizes RocksDB to manage the values stored in Flash. RocksDB employs a Log-Structured Merge-Tree (LSM-Tree) structure that makes insertions into the database very fast, but reads may require multiple storage access to retrieve a record. RocksDB is therefore ideally suited for write-heavy workloads. This aligns well with Redis on Flash, where frequently accessed data resides in RAM, and less frequently accessed data is tiered into Flash until it expires or is evicted. A typical Redis on Flash system will write frequently to Flash but read only occasionally.

The data access pattern strongly influences the performance of a Redis on Flash deployment. A very high temporal locality of reference allows most values to be served from RAM, while random access patterns stress the Flash storage layer more. The higher the Flash to RAM ratio, the more dominant the Flash storage layer becomes to database performance. A practical recommendation is to maintain a 10:1 ratio of Flash to RAM, but the ideal ratio will depend on the workload and the goals of the Redis on Flash deployment.

This document introduces a reference design that combines the ScaleFlux CSD 2000 solid-state storage drive and the Supermicro X12 FatTwin platform to deliver a high performance, low TCO platform that is uniquely suited to Redis on Flash deployments. Using a test cluster populated with reference design nodes, the performance under extreme workload corner conditions and real-world access patterns is evaluated to understand how the introduction of a Flash storage layer affects Redis performance.

The Influence of Flash on Cluster Design Considerations

The Flash storage layer in a Redis on Flash deployment resides on solid-state disks (SSDs). Due to the latency-sensitive demands of a Redis database, data center SSDs are preferable to hard disk drives. Also, the PCI-E interface will be faster than legacy SAS or SATA interfaces to the storage device.

Modern datacenter grade SSDs are offered in capacities typically ranging from 2TB to 16TB, with average capacity increasing as Flash density continues to scale. At a 10:1 ratio of Flash to RAM, few SSDs per node are required in a Redis on Flash deployment. For example, a system with 1TB of RAM may deploy 10TB of Flash storage. The small number of SSDs needed for a Redis on Flash deployment leads to several important design considerations:

1. The SSDs must individually offer very high endurance with excellent mixed workload performance since the workload will not be spread out among a large quantity of SSDs.

2. The server platform does not need to provide a large quantity of drive bays, which enables high-density solutions.

3. The increased database capacity concentrates the number of shards per node, favoring the latest generation processors with higher core counts.

In addition to addressing these key workload attributes, the capacity saved by transparent data compression can be returned to the host by expanding the logical capacity of the drive. Furthermore, capacity expansion can be performed while the drives are online without losing any existing data. This capability allows more data to be stored per dollar in a Redis on Flash cluster.

The following block diagram (Figure 1) shows where the compression and decompression take place within the Flash controller:

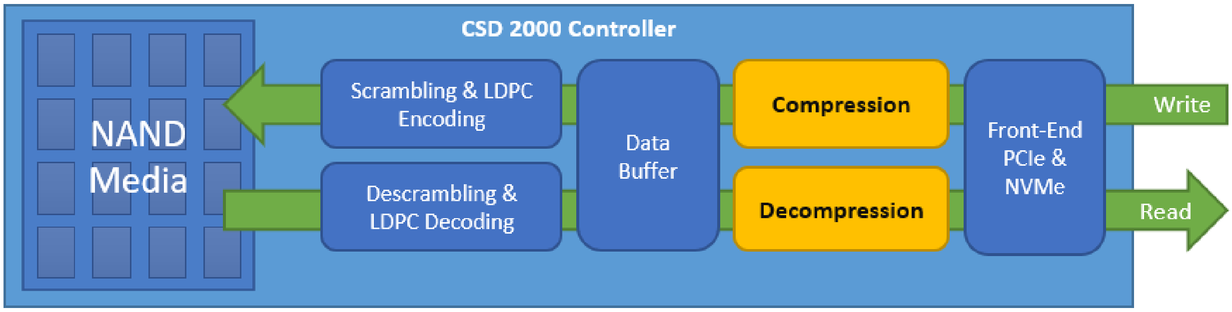


Figure 1 - CSD 2000 Block Diagram

The CSD 2000 is available in add-in card (AIC) and U.2 (2.5”) form factors. Figure 2 shows the CSD 2000 in the U.2 form factor.

Figure 2 - CSD 2000 in the U.2 Form Factor

CSD 2000 in the U.2 Form Factor

With individual Redis on Flash nodes requiring just a handful of SSDs, high-density systems are the ideal choice to maximize data center floor cost. The Supermicro X12 FatTwin provides unique half-width nodes to accommodate two nodes per rack unit. The modular left and right nodes feature redundant power supplies for high reliability. In addition, the nodes implement a tool-less design and are hot swappable for maximum serviceability.

Figure 3 - Supermicro X12 FatTwin server

Each node supports dual 3rd Gen Intel® Xeon® Scalable processors that deliver the core counts needed to support densely sharded Redis on Flash databases. Up to 2TB of DRAM are supported per node.

Redis on Flash Hardware Configuration

Racklive, a leading global data center solutions provider, configured a test cluster combining the unique capabilities of the CSD 2000 and Supermicro X12 FatTwin platform. Racklive selected the Supermicro SYS-F610P2-RTN X12 FatTwin platform with four nodes populated. Each node contains 512GB of RAM, dual Intel(R) Xeon(R) Gold 6330N CPUs (112 total v-cores), and two 3.2TB CSD 2000 SSDs configured in RAID 0 for a total Flash capacity of 6.4TB per node. All nodes are connected via a 10Gbps network. Each node runs Ubuntu 18 (Bionic Beaver) and Redis Enterprise version 6.0.20-69. Three nodes are used as database nodes, with one node reserved as the client node. All benchmarking was performed using Memtier version 1.3.0. The following figure (Figure 4) illustrates the hardware configuration:

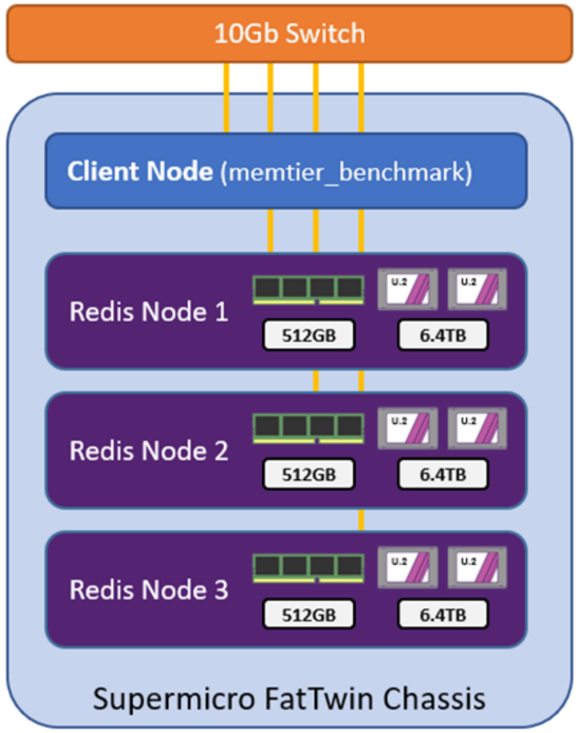


Figure 4 – Redis on Flash Hardware Configuration

Redis on Flash Software Configuration

A Redis on Flash database consisting of 1TB of RAM and 9TB of Flash capacity was configured for testing (10TB total memory limit). The database comprises 108 primary shards and 108 replica shards (216 total shards). For persistence, the fsync every second option (append-only log) was enabled. The append-only log, ephemeral data, and the RocksDB database all target mount points on the CSD 2000 RAID 0 array, which was formatted with an ext4 operating system. Note that the Linux page cache uses any free unused RAM to avoid disk access. Memory that is not used by Redis (or other host processes) will be mainly used to cache data managed by RocksDB. This improves the RAM hit rate for data stored in the Flash tier.

Initial Database Population

The database was populated with 6 billion records with a value size of 512 bytes. This object size creates a relatively large index that consumes over 60% of the available RAM reserved for Redis. This object size was chosen to place the most stress on the Flash storage layer (i.e., ensure the workload maximizes the random read IO demands on the Flash storage layer).

The following Memtier parameters were used to fill the database:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=6000000000 -n allkeys --data-size=512 --ratio=1:0 --key-pattern=P:P --cluster-
mode
```

Following the database fill, the memory profile is as follows:

- Used Memory: 5.42 TB
- Values in RAM: 310.86 M (6.56% of Values)
- Values in Flash 4.74 G
- Used RAM: 999.02GB (out of 1000GB)

The key pattern set to ‘P’ equally divides the keyspace among threads and writes keys sequentially per thread. This results in a keyspace without gaps to avoid key misses.

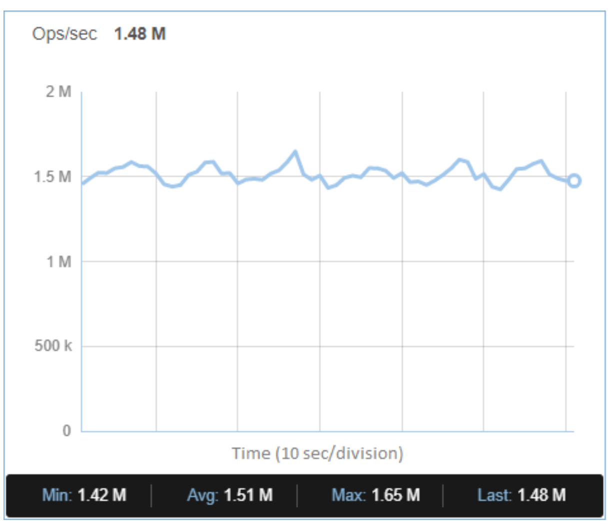
Corner Case Analysis

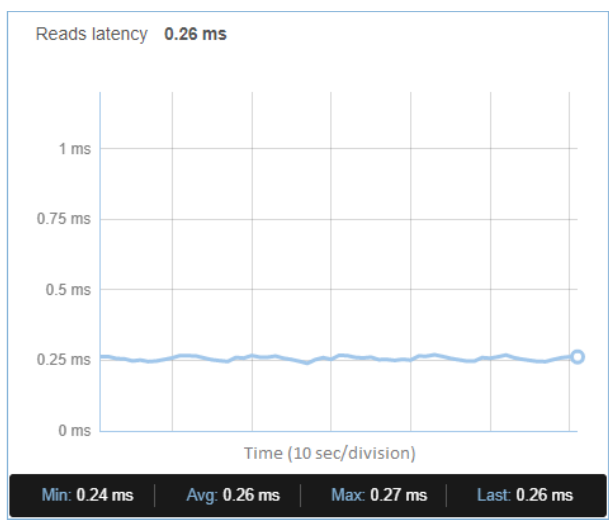
Since RAM is both higher throughput and lower latency than the Flash storage tier, the highest performance will be achieved when a workload can be served primarily from RAM. Conversely, a workload that maximizes access to the Flash storage tier will determine the lowest performance. By characterizing 100% read (GET), 100% write (SET), and 70%/30% mixed read/write (GET/SET) workloads with both maximum RAM access and maximum Flash access (corner case analysis), the performance boundary conditions can be determined.

GET - Maximum RAM Hit Ratio

The RAM hit ratio can be driven to 100% by accessing a span of the keyspace that can fit entirely into memory. Once the main memory cache tier is hot (all accessed values have been promoted into RAM), the read performance that the test cluster can achieve reaches a maximum.

This scenario results in approximately 1.5M ops/sec at a steady state at an average latency of 0.26ms (see Figure 5).





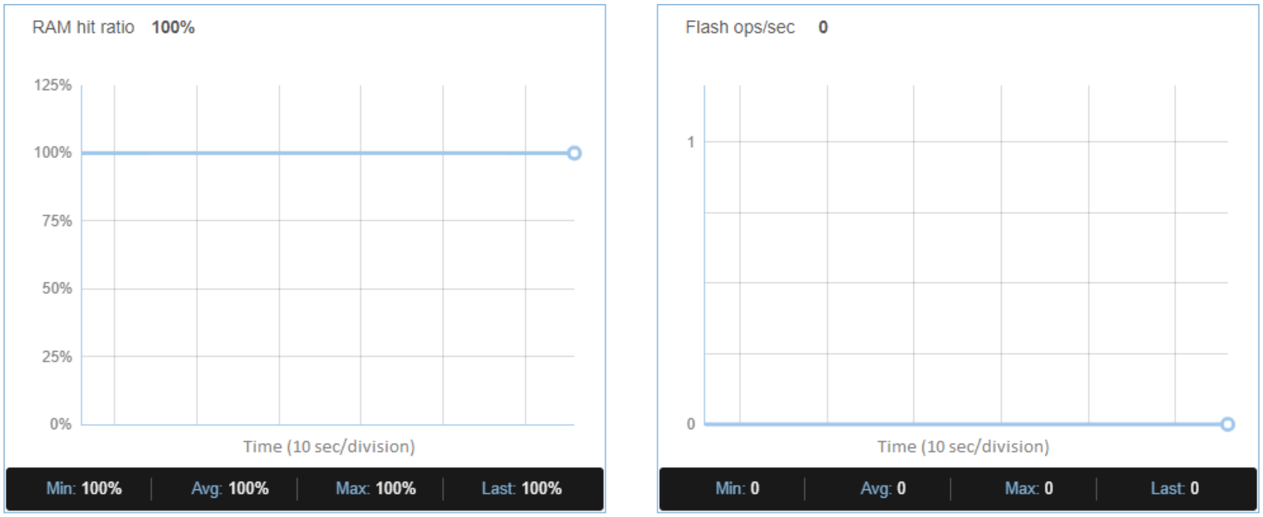


Figure 5 - GET Performance with Maximum RAM Hit Ratio

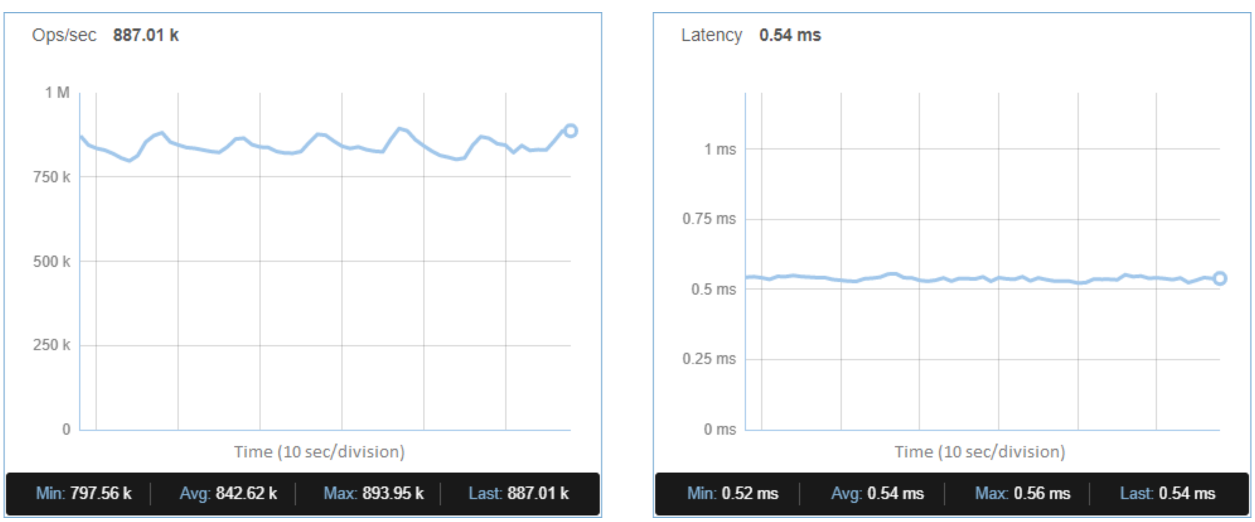
The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=100000000 --data-size=512 --ratio=0:1 --key-pattern=P:P --cluster-mode --test-
time=300 -x 5
```

GET - Minimum RAM Hit Ratio

The RAM hit ratio can be driven close to zero with a large keyspace by accessing the entire keyspace using the parallel key pattern. This also ensures that no thread promotes a key into the main memory that another thread could subsequently access.

At a steady state, this scenario results in approximately 842k ops/sec at an average latency of 0.54ms (see Figure 6).



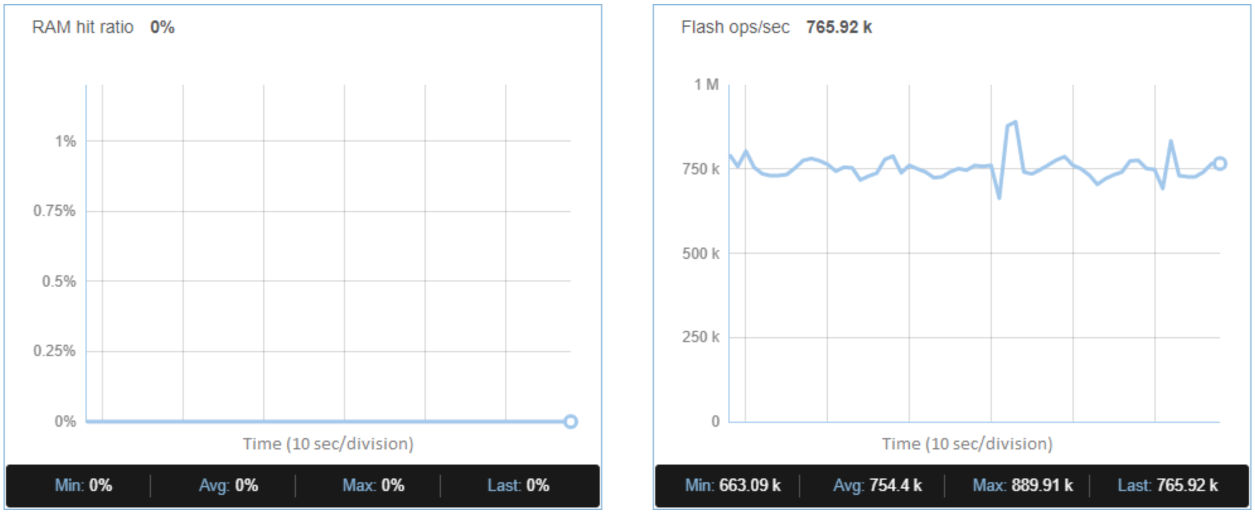


Figure 6 - GET Performance with Minimum RAM Hit Ratio

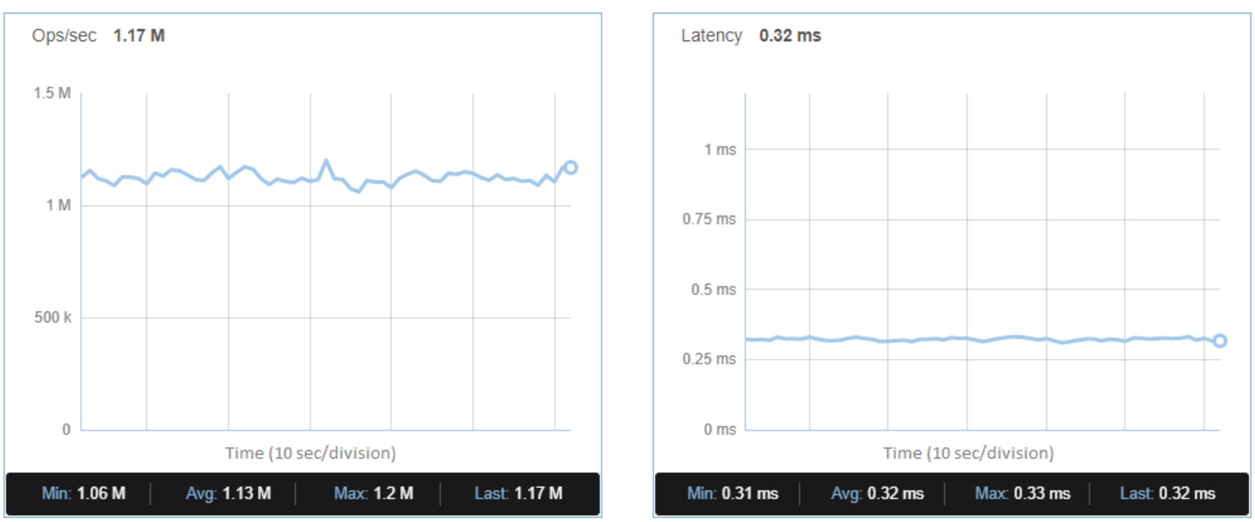
The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=6000000000 --data-size=512 --ratio=0:1 --key-pattern=P:P --cluster-mode
```

SET – Maximum Hit Ratio

Evictions to Flash can be avoided by accessing a span of the keyspace that can fit entirely into the main memory. Once the main memory cache tier is hot (all values to be updated have been promoted into RAM), the test cluster's write performance reaches a maximum.

At a steady state, this scenario results in approximately 1.17M ops/sec at an average latency of 0.32ms (see Figure 7).



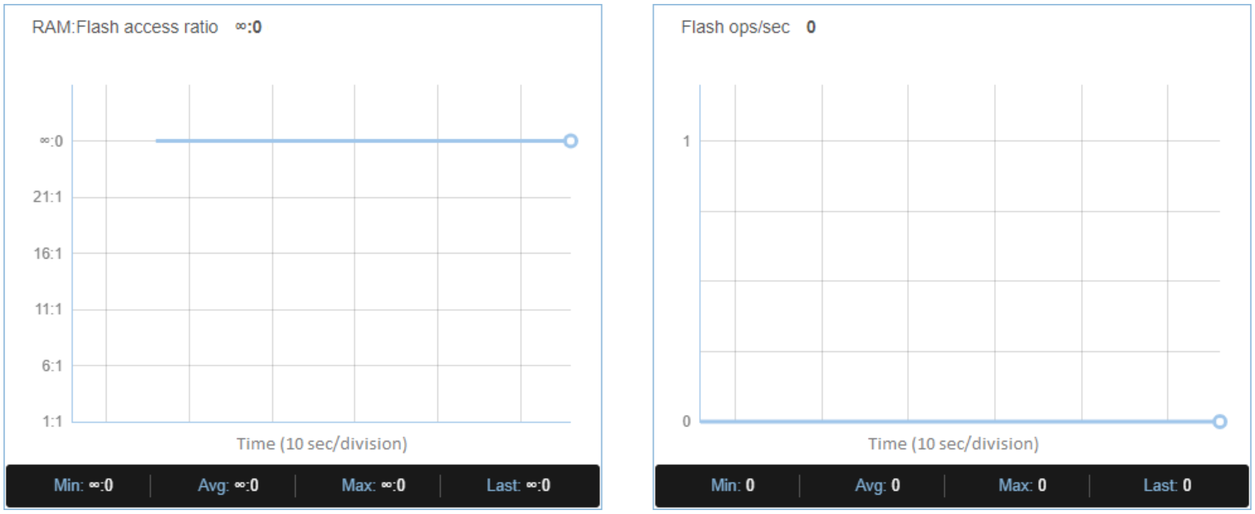


Figure 7 - SET Performance with Maximum RAM Hit Ratio

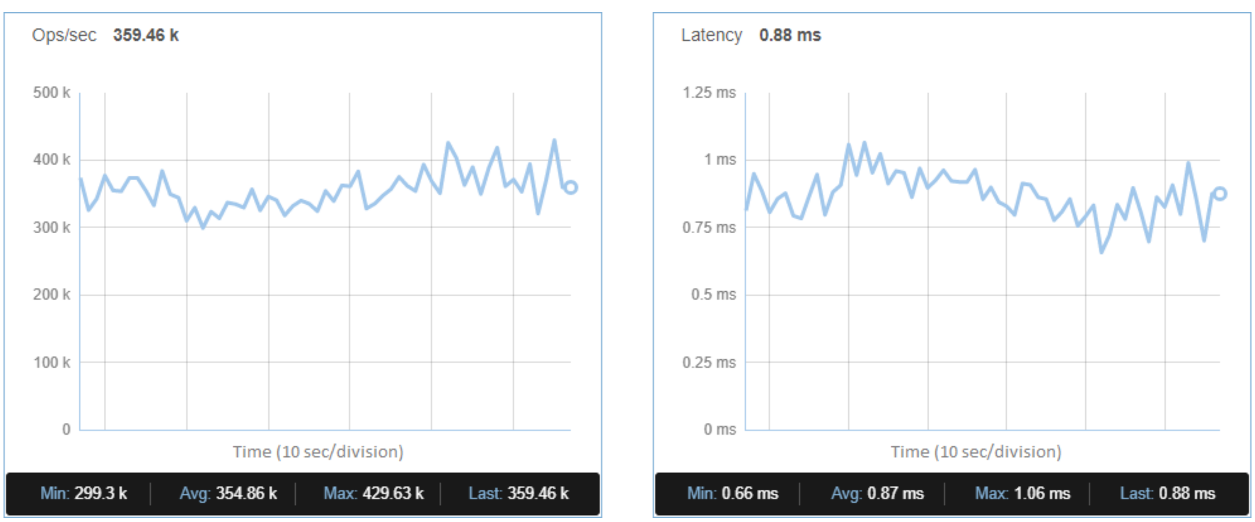
The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=150000000 -n allkeys --data-size=512 --ratio=1:0 --key-pattern=P:P --cluster-
mode -x 4
```

SET – Minimum Hit Ratio

With a large keyspace, all SET operations can result in an eviction to Flash. This places the worst-case load on the Flash storage layer.

At a steady state, this scenario results in approximately 359k ops/sec at an average latency of 0.88ms (see Figure 8).



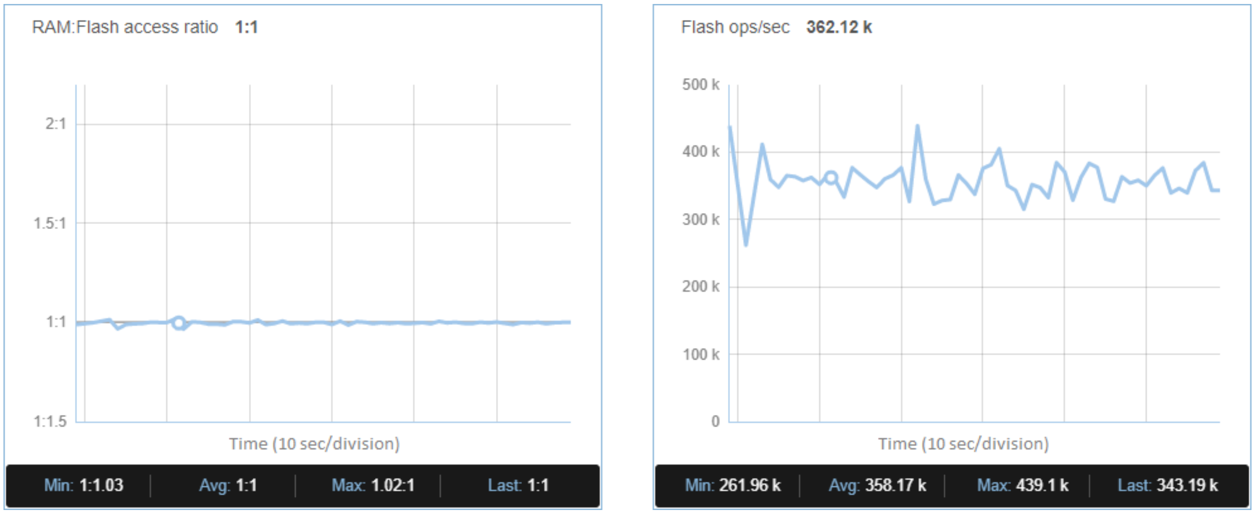


Figure 8 - SET Performance with Minimum RAM Hit Ratio

The 1:1 RAM to Flash access ratio reflects the nature of Redis on Flash, where SET operations are stored in RAM, and an equal number of older values are de-tiered to the Flash storage layer.

The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-maximum=6000000000 -n allkeys --data-size=512 --ratio=1:0 --key-pattern=P:P --cluster-mode
```

70/30 GET/SET – Maximum Hit Ratio

As with the pure GET and SET corner cases, exercising a keyspace that fits within the main memory results in maximum performance.

At a steady state, this scenario results in approximately 1.59M ops/sec at an average latency of 0.25ms (see Figure 9).

Figure 9 - Mixed GET/SET Performance with Maximum RAM Hit Ratio

The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=150000000 -n allkeys --data-size=512 --ratio=3:7 --key-pattern=P:P --cluster-
mode -x 4
```

70/30 GET/SET – Minimum Hit Ratio

As with the pure GET and SET corner cases, exercising the entire keyspace maximizes the use of the Flash storage layer.

At a steady state, this scenario results in approximately 706k ops/sec at an average latency of 0.58ms (see Figure 10).

Figure 10 - Mixed GET/SET Performance with Minimum RAM Hit Ratio

The above data was collected using the following Memtier parameters:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 64 --key-
maximum=150000000 -n allkeys --data-size=512 --ratio=3:7 --key-pattern=P:P --cluster-
mode
```

Corner Case Testing Conclusions

Flash latency is three orders of magnitude higher than RAM; nonetheless, running the workloads exclusively from Flash achieves 31% of RAM performance for write (SET) and 56% of RAM performance for read (GET). In both cases, the average latency remains below a 1ms threshold. The Flash storage layer provides an appreciable level of performance under worst-case conditions while extending database capacity by nearly 10x.

The following table (Table 1) summarizes the performance deltas for each corner test:

Corner Case	Maximum RAM Hit Ratio		Maximum Flash Hit Ratio		Flash Performance Delta	
	kops/sec	Avg. Latency (ms)	kops/sec	Avg. Latency (ms)	kops/sec	Avg. Latency
100% GET	1500	0.26	842	0.54	56%	208%
100% SET	1170	0.32	359	0.88	31%	275%
70/30 G/S	1630	0.25	706	0.58	43%	232%

Table 1 - Summary of Corner Case Test Results

Summary of Corner Case Test Results

While corner case testing establishes the upper and lower bounds of performance, real world access patterns are expected to demonstrate a high degree of temporal locality of reference. As a result, there will be a strong access bias for more recent records available in RAM.

The Gaussian key pattern option in Memtier can be used to model such access patterns. For the tests described in this section, the Gaussian access pattern under different standard deviation (σ) values is used to characterize the performance.

The following plot (Figure 11) illustrates the standard deviation values that were tested:

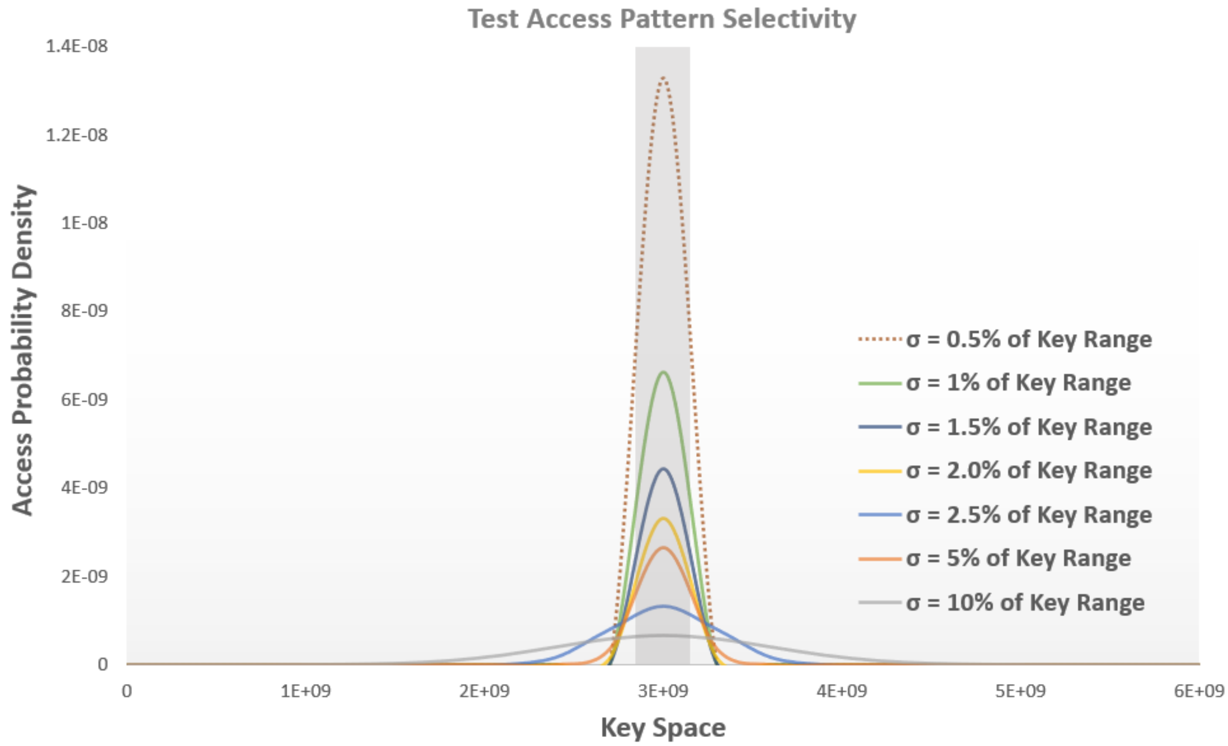


Figure 11 - Test Access Pattern Distributions

The gray box is a visual aid representing the maximum quantity of values stored in RAM. It does not indicate the actual key range that will be present in RAM at any given time.

The template for the Memtier commands used to test different standard deviation values is as follows:

```
$ memtier_benchmark -s <Node 1 IP> -p <DB Port> --pipeline=8 -c 1 -t 8 --key-maximum=6000000000 -n allkeys --data-size=512 --ratio=3:7 --key-pattern=G:G --cluster-mode --key-stddev=<Set per Desired Standard Deviation> --distinct-client-seed
```

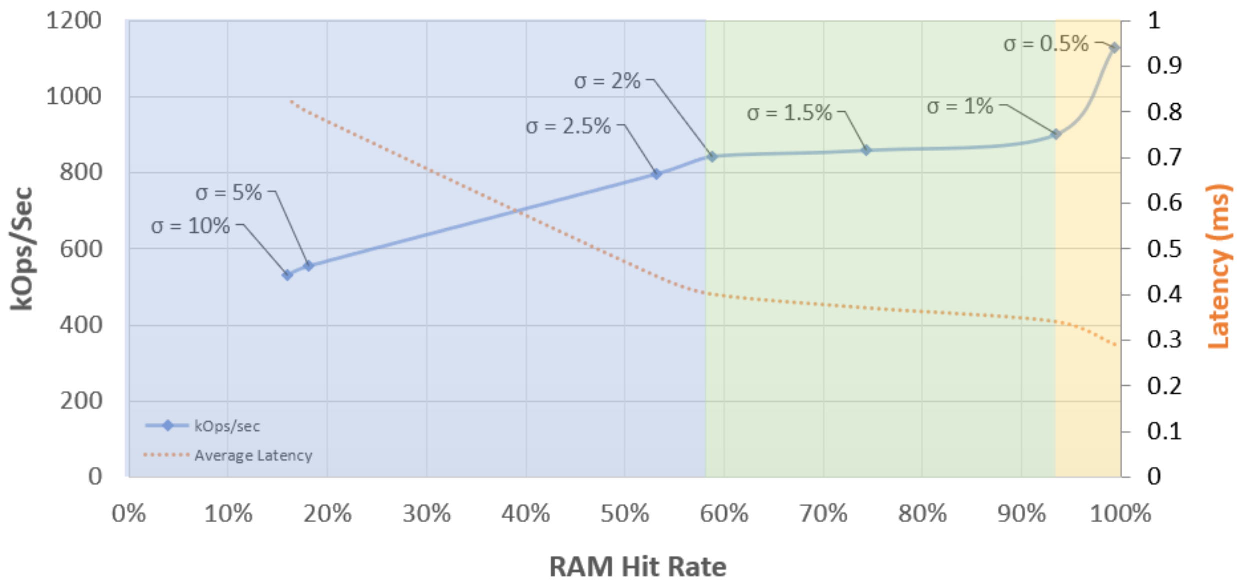


Figure 12 - Performance vs. RAM Hit Rate

Performance remains highly consistent between a hit ratio of approximately 60% and 90%. This remarkably large band accommodates 40% of access, reaching the Flash storage tier with a high latency consistency. As the hit ratio decreases below 60%, the Flash layer becomes saturated. As the hit ratio exceeds 90%, performance becomes increasingly RAM dominated; however, note that while there are gains in the number of operations per second, the average latency is not significantly improved compared to hit ratios down to 60%.

A Closer Look at Flash Throughput

Figure 13 shows the instantaneous read throughput collected over a three-minute interval at three different sigma levels.

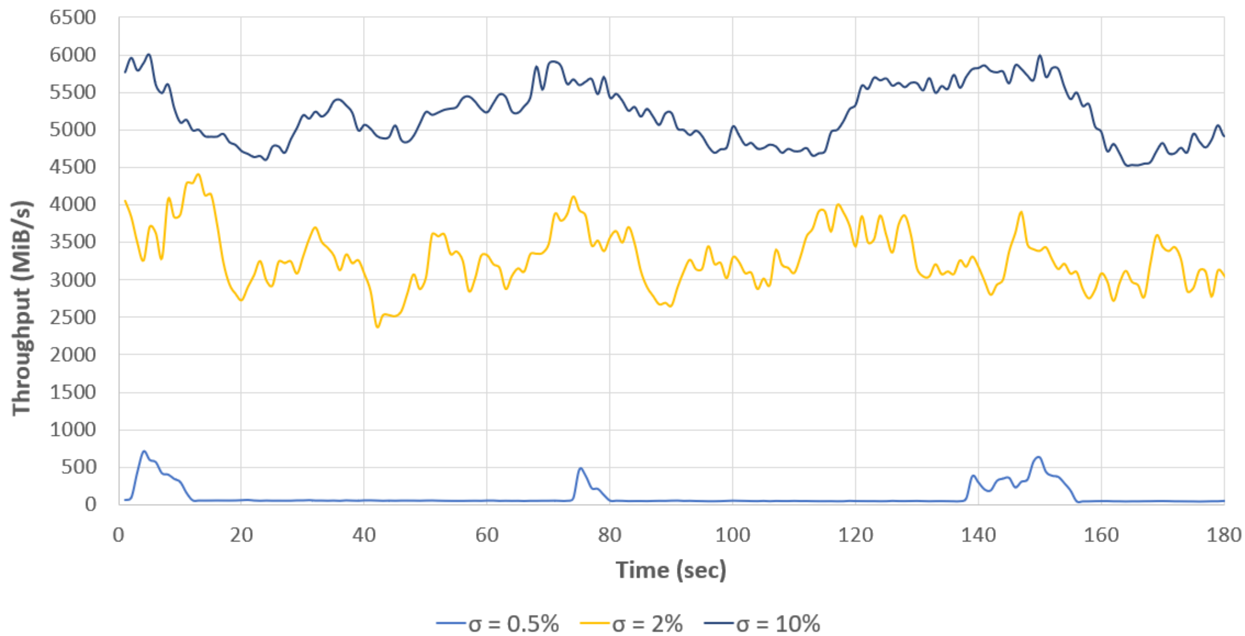


Figure 13 - Flash Read Throughput over Time

The read workload from Flash is relatively steady. At the lowest RAM hit ratio tested (σ = 10% of the key range), the read throughput peaks to the maximum limit provided by the RAID0 array (6GiB/s). The write workload to Flash is much more mixed. It consists of persistence data, ephemeral data, and the RocksDB workload. Persistence data is flushed every second and produces a baseline workload. This can be most easily observed in the case where σ = 0.5 of the key range. On the other hand, the RocksDB workload is characterized by periodic bursts corresponding to table flushing from RAM (see Figure 14).

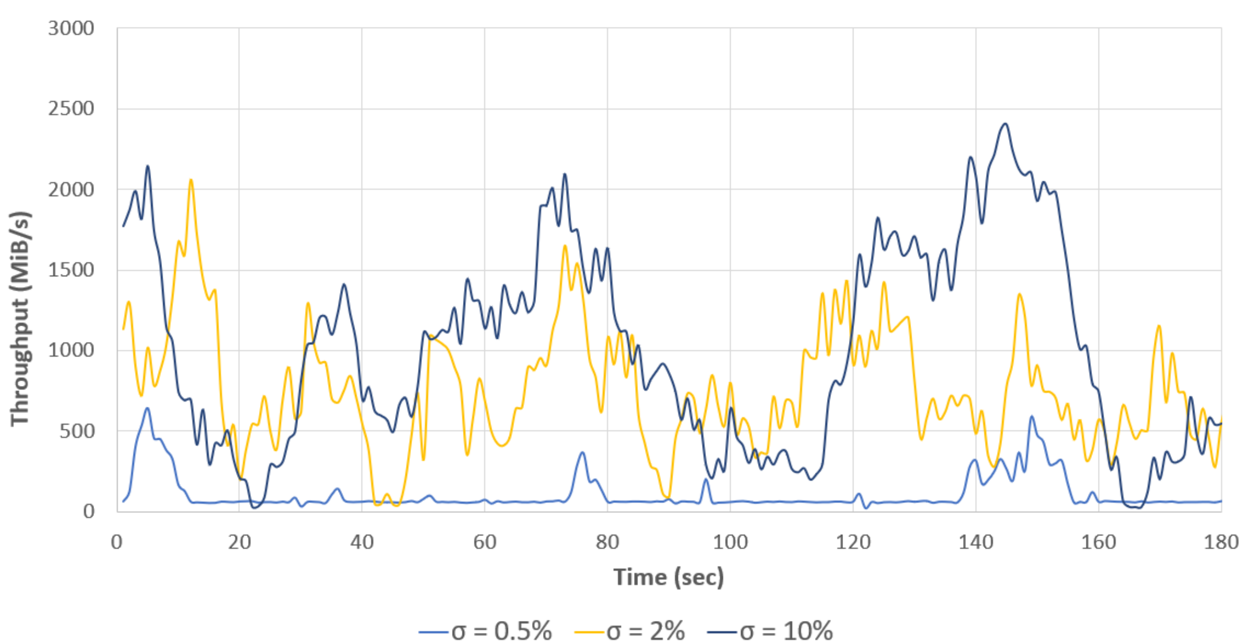


Figure 14 - Flash Write Throughput over Time

Flash Endurance

Flash devices are rated for a total amount of write activity (or endurance) expressed in either drive writes per day (DWPD) or in total bytes written (TBW). With the write heavy workload of RocksDB, it is essential to characterize the Flash storage layer's performance and its expected life.

The following table (Table 2) shows the expected service life as a function of average write throughput and the PBW rating:

Average MiB/s	TBW Rating					
	10 PBW	15 PBW	20 PBW	30 PBW	35 PBW	40 PBW
100	3.02 Years	4.54 Years	6.05 Years	9.07 Years	10.58 Years	12.10 Years
200	1.51 Years	2.27 Years	3.02 Years	4.54 Years	5.29 Years	6.05 Years
300	1.01 Years	1.51 Years	2.02 Years	3.02 Years	3.53 Years	4.03 Years
400	0.76 Years	1.13 Years	1.51 Years	2.27 Years	2.65 Years	3.02 Years
500	0.60 Years	0.91 Years	1.21 Years	1.81 Years	2.12 Years	2.42 Years

Table 2 – Drive Life vs. Write Throughput

Values in green indicate the PBW ratings required to avoid wear-out within a three-year warranty term for a drive.

Conclusion

Combining the ScaleFlux CSD 2000 with the Supermicro X12 FatTwin platform creates a compelling platform for Redis on Flash deployments. Benchmarking data showed consistent, low-latency performance with RAM hit ratios as low as 60% using just two CSD 2000 devices per node. In addition, the half width architecture of the Supermicro X12 FatTwin platform slashes physical space requirements, while transparent datapath compression featured in the CSD 2000 cuts the storage space requirements – all while addressing the key workload concerns of a Redis on Flash deployment: write endurance and read performance.

