WHITE PAPER

Building Large Databases with Redis on Flash

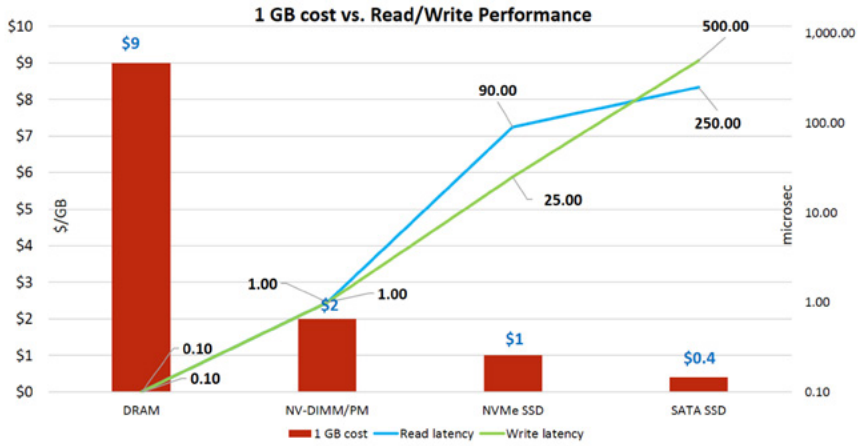
Cihan Biyikoglu, VP, Product Management, Redis

CONTENTS

Why Redis on Flash?

Redis is known for its sub-millisecond latencies and high throughput. For many interactive applications, responsiveness is key to engaging and fluid experiences. However, keeping a large amount of data in RAM with Redis can be expensive. Today many applications either pay a premium for storing sizable datasets in RAM with Redis or they limit Redis database use to the most valuable data and augment Redis with disk-based relational or NoSQL databases.

With new advances in hardware, flash technology (SSDs based on SATA or NVMe) provides a great alternative for low latency read/writes and comes at a cost 10-20 times cheaper than RAM (on a per GB basis). RAM continues to dominate with lowest latency data access guarantees. However, flash technology is catching up fast with NV-DIMM and NVMe SSDs from major vendors.



Redis on Flash technology combines RAM and flash to store large data sets in Redis with much lower cost per GB. With Redis on Flash, you can extend RAM onto Flash memory and keep larger data sets in Redis, all without losing Redis' performance advantage.

To provide the best data access performance, Redis on Flash uses smart data placement, storing frequently accessed data in RAM and less frequently accessed data in flash. The cost savings to store "10TB in RAM" vs. "2TB RAM + 8TB in a Flash drive" (SATA or NVMe-based SSDs) can often be more than 80%.

Redis on Flash Architecture

A Redis Enterprise cluster is composed of identical nodes that are deployed within a data center or stretched across local availability zones. Redis Enterprise architecture is made up of a management path (depicted in the blue layer in Figure 1 below) and data access path (depicted in the red layer in Figure 1 below).

- The **Management path** includes the cluster manager, proxy and secure REST API/UI for programmatic administration. In short, the cluster manager is responsible for orchestrating the cluster and the placement of database shards, as well as detecting and mitigating failures. Proxy helps scale connection management.
- The **Data Access path** is composed of master and slave Redis shards. Clients perform data operations on the master shard. Master shards maintain slave shards using the in-memory replication for protection against failures that may render the master shard inaccessible.

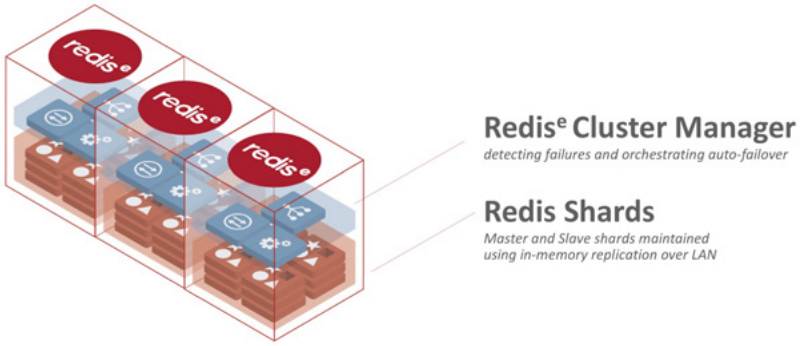


Figure 1. *Redis Enterprise nodes, with blue tiles representing the management path and red tiles representing the data access path with Redis as the shards.*

High Availability with Redis Enterprise

Redis Enterprise uses in-memory replication to maintain master and slave replicas. Redis Enterprise comes with various watchdogs that detect and protect against many failure types. In node, network and process failures that render the master replica inaccessible, Redis Enterprise automatically promotes the slave replica to be a master replica and redirects the client connection transparently to the new master replica.

Besides the intra-cluster replication, Redis Enterprise also has built-in WAN-based replication for Redis deployments across multiple data centers. WAN-based replication mechanisms in Redis Enterprise are designed to protect against total data center (or wider) network failures.

All the high availability capabilities operate the same in Redis on Flash as well. You can find additional details on Redis Enterprise high availability capabilities in the references section.

Scaling Databases with Redis on Flash

Each Redis Enterprise cluster can contain multiple databases. In Redis, databases represent data that belong to a single application, tenant or microservice. Redis Enterprise is built to scale to hundreds of databases per cluster to provide flexible and efficient multi-tenancy models.

Each database can contain few or many Redis shards. Sharding is transparent to Redis applications. Master shards in the database process data operations for a given subset of keys. The number of shards per database is configurable and depend on the throughput needs of the applications. Databases in Redis Enterprise can be resharded into more Redis shards to scale throughput while maintaining sub-millisecond latencies. Re-sharding is performed without downtime.

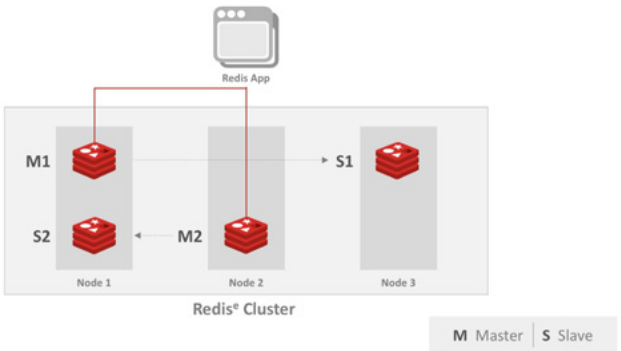


Figure 2. *Redis Enterprise places master (M) and slave (S) replicas in separate nodes, racks and zones and use in-memory replication to protect data against failures.*

In Redis Enterprise, each database has a quota of RAM. The quota cannot exceed the limits of the RAM available on the node. However, with Redis on Flash, RAM is extended to the local flash drive (SATA, NVMe SSDs etc). The total quota of the database can take advantage of both RAM and flash drive. The administrator can choose the RAM vs Flash ratio and adjust that at any moment in the lifetime of the database without downtime.

With Redis on Flash, instead of storing all keys and data for a given shard in RAM, less frequently accessed values are pushed to flash. If applications need to access a value that is in flash, Redis Enterprise automatically brings the value into RAM. Depending on the flash hardware in use, applications experience slightly higher latency when bringing values back into RAM from flash. However subsequent accesses to the same value is fast, once the value is in RAM.

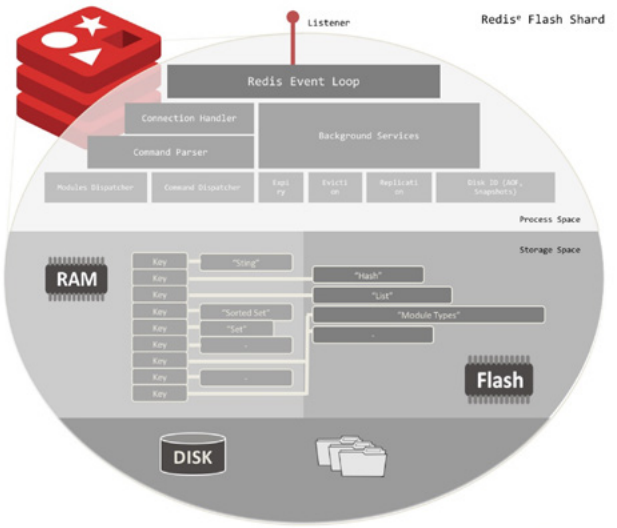


Figure 3. *Redis Enterprise Flash shards with process, memory and disk storage components. Redis Enterprise Flash uses both RAM and Flash for keeping data. RAM store all keys and some values. As the RAM fills up, less frequently used values are moved to flash (NVMe or SATA based SSDs).*

In the majority of database workloads, there isn't a uniform distribution of load. That is to say, some keys in the system gets accessed more often than others. The frequently accessed portion of the data is called the "hot-working-set." As the hot-working-set changes over time and new data arrives, Redis Enterprise Flash adapts to the new workload. Redis Enterprise Flash has a background task that ejects less frequently used values to flash in order to adapt and maintain a healthy dose of free space for new incoming operations.

It is important to note that even though values get ejected to flash, all keys and metadata stay in RAM. Keys are typically smaller in size than values. Many Redis commands require access to keys without requiring access to the value. Keeping the full list of keys in RAM ensures many operations can be executed without any penalty of value retrieval from flash. Here is an example where keeping keys in RAM results in substantial savings:

Imagine the "SET" command with NX option, which causes the SET operation to fail if the key already exists (for a detailed reference on SET with NX see the reference page here). In this case, having keys in RAM is a huge benefit to keep latency of the operations low. With all keys stored in RAM, it is easy to check if the key already exists before inserting the new key. This is just one example of which there are many.

Data Durability with Redis Enterprise

Redis Enterprise Flash uses a flash drive as a RAM extension. At bootstrap of the database, Redis Enterprise Flash expects and ensures that both RAM and Flash drive are completely empty. Once the engine is started, RAM+flash is populated from the durable copy of data (disk or another replica). When using Redis Enterprise Flash, you can use either of

Redis Enterprise's two durability options:

- **Disk-based durability:** Redis Enterprise still maintains a durable copy on disk. Just like disk-based systems, this IO path is placed on a slower and more durable network-attached storage device. Redis databases provide tunable options to maintain this durable copy and keep it up to date, from frequent periodic writes all the way to per-operation writes.
- **Replication-based durability:** Redis Enterprise also maintain a replica--a slave shard--for durability. Replication-based durability protects against node, rack or zone failures and provides better write performance than network-attached storage writes. This means that in the event of an unplanned interruption, it is likely that your replica is more up to date than your durable copy on disk. To take full advantage of the replicated-durability, Redis provides the WAIT command. WAIT ensures that a write can wait for acknowledgement until multiple replicas confirm that write. This ensures that a write confirmed with WAIT on replicas will be durable even if a node catches on fire and never comes back to the cluster.

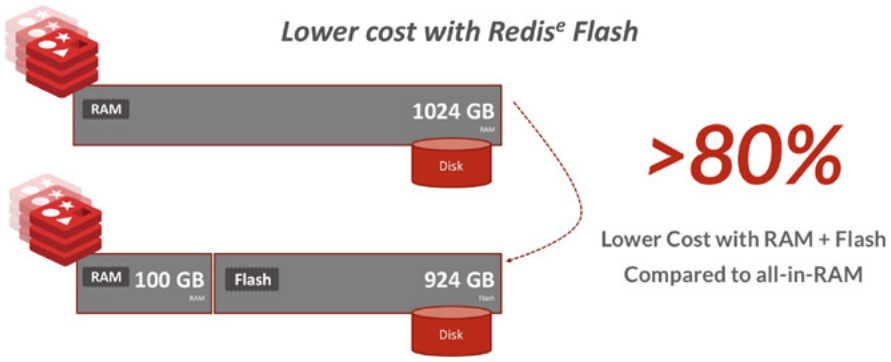
"Buffer Cache" vs The "RAM Extension" Approach

The smart data placement in Redis Enterprise Flash, which brings values from flash into RAM based on working set, is similar to disk-based database systems and the "Cache miss" on the buffer cache of the database. However, similarities between disk-based databases and the RAM extension method used in Redis Enterprise Flash end there. The IO patterns used in Redis Enterprise Flash are much more efficient than those of a disk-based system. Here are some of the differences between the two:

- **Hot Value Handling:** Many application workloads perform repeated writes to a set of "hot" keys in a short period of time, such as when the keys belong to an active piece of data. For example, imagine repeated updates by an app to its database, tracking a current shopper's state on a site as the shopper views various products. Disk-based databases perform these writes both in RAM and on disk to persist the changes each time. However, updates to data in RAM are not sent to flash in Redis Enterprise Flash. The RAM extension approach used by Redis Enterprise Flash does not require any writes to flash under repeated writes to "hot" keys, unless the value gets ejected to flash. Remember that active values don't get ejected to flash and mostly stay in RAM, so the repeated updates to the active shopper's keys simply happen in RAM and do not require flash writes. This ensure that the IO bandwidth of the Flash drive is only used for ejections to Flash.
- **Exploiting Ephemeral Storage:** The cloud architecture in public or private clouds typically comes with two types of storage, faster ephemeral storage and slower durable network-attached storage. Disk-based databases require their writes to persist all the way to disk for every write. Thus you are required to use the persisted network-attached storage. However, Redis Enterprise Flash treats flash memory as a RAM extension, thus it can fully take advantage of local, fast ephemeral storage.
- **Write Amplification:** Disk-based databases depend on disk writes for durability. Each write to disk in disk-based databases is typically done through a redo-log (RL) or a write-ahead-log (WAL) before the actual values are updated on storage. Redis Enterprise Flash uses RocksDB to manage the flash drive access. The call sequence to RocksDB with Redis Enterprise Flash does not need to maintain these additional WALs. Write amplification measures the number of IO operations that any single read/write causes. Due to the logged writes, disk-based databases end up with much higher write amplification. You can read more about RocksDB and various IO amplification effects here.
- **Advances in HW with Persistent Memory:** The techniques used in creating Redis Enterprise Flash are based on the new direction in memory technology. As persistent memory is introduced into the compute architecture such as Intel's 3DXPoint, the idea behind these technologies is to allow the application to decide which part of the data will be kept in RAM and which will use Flash/Nand in-order to maximize performance at the optimal cost. Redis Enterprise Flash was designed to exploit these benefits.

Redis Enterprise Flash Use Cases

You can use Redis Enterprise to build responsive interactive applications. With Redis Enterprise Flash, you can extend RAM onto Flash drive and keep larger data sets with lower cost infrastructure while retaining Redis' performance advantage. The price difference between RAM vs Flash can be 10x to 20x per GB. This can translate to savings of over 80% in infrastructure costs.



DRAM, flash, and SSD memory are becoming increasingly more affordable. Redis Enterprise Flash combines the speed of RAM and the affordability of flash to deliver real-time data access with sub-millisecond latency, even with large data sets. The common use cases where Redis Enterprise Flash makes for a perfect fit are:

Real-Time Analytics

Use the power of Redis and the reliability and scalability of Redis Enterprise Flash to deliver high-velocity analytics such as behavior-based personalization; recent purchases or trending items in e-Commerce and retail applications; leaderboards and top scorers in gaming applications; real-time fraud detection; real-time ad placement, etc.

Fast Data Ingest

Use the high-performance data structures of Redis to continuously ingest high velocity data--such as millions of events from websites or IoT devices--with very little hardware.

Time-Series Data

Redis Sorted Sets accelerate processing and analysis of time-series data by orders of magnitude compared to other disk-based key value stores. Use Redis Enterprise Flash to gain a high performance, highly available, time-series-processing engine.

Metering and Traffic Shaping

Redis offers a rich set of data structures, atomic counters and features (such as "time-to-live")that will help you count and regulate items. Redis Enterprise Flash enables you to enforce traffic shaping and rate limiting on extremely large data sets, while maintaining great price efficiency.

References

- Getting Started with Redis Enterprise Pack
  - Redis Enterprise Flash
- Redis Enterprise Flash
  - Overview
  - Best Practices and Considerations
- Redis Enterprise Architecture
  - Clustering Architecture

700 E El Camino Real, Suite 250
Mountain View, CA 94040
(415) 930-9666
redis.com