WHITE PAPER

Do You Really Know Redis?

Leena Joshi, VP Product Marketing, Redis Labs

CONTENTS

Introduction

Redis is tremendously popular among developers all over the world, and chances are you use many applications that utilize Redis, under the covers, in one way or another. Developers use it for its simplicity, high performance and dynamic community. This paper attempts to dispel some common misconceptions, clarify exactly why Redis is so popular and illuminate why it is evolving into an all-purpose preferred database.

Definition

MLet's look at the Redis definition from redis.io:

> *Redis is an open source (BSD licensed), in-memory **data structure store**, used as database, cache and message broker. It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.*

Let's start with the **"database, cache and message broker"** usage. How do these elements differ from each other?

> *A **database** is a collection of information that is organized so that it can be easily accessed, managed, and updated.*[1]

> *In computing, a **cache** is a component that stores data so future requests for that data can be served faster; the data stored in a **cache** might be the result of an earlier computation, or the duplicate of data stored elsewhere.*[2]

> ***Message broker** is an intermediary program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver.*[3]

Is it a database? A cache? Or message broker?

Redis starts off frequently as a cache, meaning a place to store results of previous queries or interim computation results while the primary data storage is a SQL or NoSQL disk based database. Redis has several built in mechanisms which make it the best choice for addressing common caching requirements.

Data persistence and high availability in Redis set it apart from a common cache. We find that customers frequently use Redis to store high velocity data that is not stored anywhere else - i.e. Redis is used as a primary database for data that requires rapid processing. Another really common scenario is Redis being used as user-facing database, serving customers in real-time, while a secondary (usually slower) database is used for recording the transaction history.

The key difference between this approach and a cache scenario here is that Redis is the only place where the high velocity data is available. This is possible with Redis and not other caching technologies because Redis can persist its data to disk and can be made highly available through in-memory replication and auto-failover.

Example: A large bank that offers Apple Pay services uses Redis to store parameters presented during Apple Pay transactions - these parameters are provided per session, stashed in Redis and constantly accessed during each extremely high speed session- they are not stored elsewhere. This represents Redis' use as a database.

Redis is also used as a message broker following the publish-subscribe paradigm, where published messages are characterized into channels. Subscribers only receive messages that are of their expressed interest. While it is not a single purpose message broker like Apache Kafka or RabbitMQ, the reason to use Redis in this way is often that in addition to being a message broker, Redis offers data structures (particularly the List data structure) and notification mechanisms (i.e. publish,

[1]http://itclass.heinz.cmu.edu/itppmweb03/group11/project/Glossary/GlossaryIndex.htm
[2]https://en.wikipedia.org/wiki/Cache_(computing)
[3]https://en.wikipedia.org/wiki/Message_broker

subscribe) that support the use case and provide a performance boost. This implementation is also the backbone for many popular job management system like Sidekiq, Resque, Celery and others.

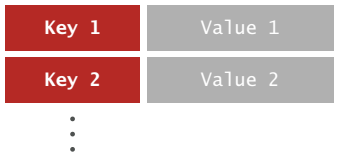
Customers use Redis in all three ways – database, cache and message broker. This multi-purpose nature of Redis makes it much more powerful than other in-memory simple key value stores but it is not the only characteristic that makes Redis unique.

The importance of data structures

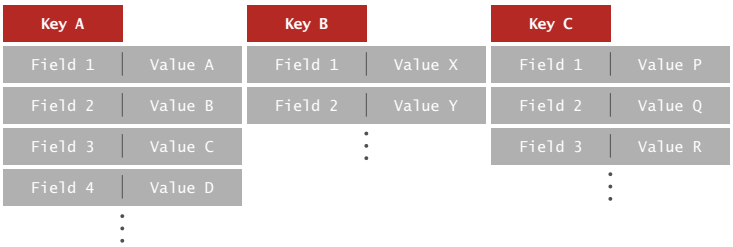
Data structures are like "Lego" building blocks; helping developers achieve specific functionality with the least amount of complexity, least overhead on the network and least latency because operations are executed extremely efficiently in memory, right next to where the data is stored.

Data structures set Redis apart from other key/value stores and NoSQLs in terms of versatility and flexibility. Redis data structures include:

Strings: The most basic data structure found in many key value stores. Redis allows for direct operations on the data—such as incrementing numeric values and manipulating string ranges—in a highly efficient and atomic manner.

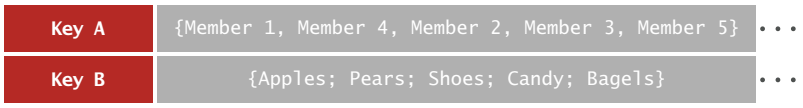


Hashes: Hashes (a.k.a maps, dictionaries and associative arrays) are elementary data structures that are essentially collections of field-value pairs. In a sense, each Hash is a mini-Redis inside Redis itself, and this is where Redis starts breaking away from other K/V stores as well as other NoSQLs. Hashes allow for fast access and manipulation of individual values within the collection and provide 15 built-in operations. Hashes are an extremely flexible data structure that can be used to model a wide range of structured data requirements such as user sessions and profiles, JSON documents or even relational tabular-like data.

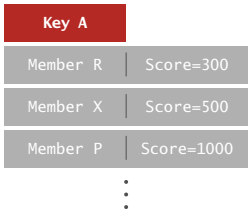


Lists: Redis' Lists are made of String elements that are linked one to another, much like the links of a chain. Lists provide fast constant access time to their ends for adding and removing elements (even with millions of items) and retain their internal order. Queues are a popular use case for Lists, in which new elements, representing jobs or messages, are added (pushed) to one end of the list and existing elements are removed (popped) from the same or other end. Multiple blocking commands (example: 'wait until timeout or until something is pushed to the queue') allow easy facilitation of asynchronous, producer-consumers patterns.

Sets: A Set is an unordered collection of unique strings. On top of adding to and removing elements from Sets, Redis can also perform all types of operations on them, namely: intersect, difference and union. This makes Sets particularly useful for de-duplicating data, representing the properties of data entities and answering questions about membership (of the elements in them). Among the uses for Sets are making recommendations, identifying similar items, finding common purchases, interests or friends.



Sorted Sets: Sorted Sets, as their name implies, order their members by a score. Retrieving members by score ranges is trivial with Sorted Set operations and make them a natural fit for time-series data, real-time bid management, purchases by order amounts, most viewed articles, top scores etc. Sorted Sets are built in Redis with mechanisms that provide high performance sorting. Redis Sorted sets can also be used as a building block for a search engine.



Geospatial indices: These are a specialized form of Sorted Sets, in which each members' score encodes a geographical coordinate. Redis' ultra-fast data structures are ideal for performing searches within an area, such as with navigation and location-based services.

Bit arrays and fields: Bit arrays and fields offer a rather unique set of operations that are useful when you have millions of things to be counted but want to limit memory spent—think of tracking online status of hundreds of thousands of users with only as many bits—and using operations like BITCOUNT to report on how many users are online at any given time. In IoT scenarios, where there are millions of data points, and several thousands of things to be counted, bit arrays are an extremely memory efficient way to implement such counters.

Hyperloglogs: HyperLogLog is a probabilistic data structure used to estimate the count of unique things. This data structure trades accuracy with extreme memory and processing power savings.

As you can see, these data structures are optimized for specific application scenarios and functionality - for these scenarios, Redis is unparalleled in execution efficiency and performance when compared to any other database whether SQL, NoSQL, in-memory or disk-based.

The data structures are also the reason one sees user reviews like this for Redis:

"The robust data structures that Redis offers are by far my favorite feature. It can be used as a simple key/value store like you would do with Memcached but also allows you to create hash tables to simulate more Mongo-like features as well as lists and sorted sets that can be used for queues, chat systems and leaderboards (basically anything you want sorted). Also, the option to flush to disk and have persistence is pretty awesome."

"I use Redis for a series of things, specifically a chat system and leaderboards for tracking site activity. I have also used Redis a ton in front of MySQL to serve as a more direct index in scenarios where queries are under-performing and the tables are so larger (50m+ rows) that trying to reindex means a chunk of downtime. I also use Redis for user sessions. I am also starting to migrate away from Memcached for basic key/value stores in front of MySQL."

For operations that are more complex, Redis supports Lua scripting which allows for complex computations, elaborate queries and more.

When compared to other key/value stores, Redis is orders of magnitude more powerful due to its data structures, persistence, high availability and scalability.

When you compare Redis to other NoSQLs, for example, MongoDB with its document data model, its representation of JSON documents is similar to the hash structure in Redis. Cassandra, with its wide column data store can be thought of as a special case of the sorted set/hash data structure too - the notable difference being that operations on those data models are not executed in-memory, or in-database and impose significant performance penalties.

How Customers Use Redis

The below use cases show how Redis is used as a database, cache and message broker – in operational and analytics uses.

Operational and analytics database: One of the most common use cases of Redis is session storage. Cooladata, a managed event based data warehouse in the cloud, tracks user events across hundreds of thousands of users on popular websites. This event stream firehose would require too many instances of disk based SQL or NoSQL databases to handle updates and inserts. Redis is used to sessionize this data and generate real time analytics on user sessions. A summary of sessions is then stored in MongoDB for historical recording and HDFS for batch analysis. In this scenario, Redis is the main first responder database and provider of real-time analytics.

Transactions, analytics and message delivery: Xignite, the market data cloud provider, uses Redis to provide lightning fast access to market data, power complex computations and for its publish/subscribe interfaces.

Time-series data analysis and delivery: An options trading platform-as-a-service provider uses Redis to deliver rapidly changing price/currency pairs at extremely high velocities (hundreds of price-currency pairs, each one changing >10 times/second delivered to several hundred subscriber systems with <1ms latencies). Redis forms the highly available and scalable backbone of this update and delivery.

Stance, a pioneer of the modern retail shopping experience, uses Redis as its inventory notification provider to thousands of requesting clients with sub-millisecond latencies.

Operational database: A large telecommunications provider uses Redis as its main NoSQL database to power its CDR/Billing application that requires over 5 million operations per second at <5 msec latencies.

A content delivery and advertising platform provider uses Redis for real time display of content to its subscriber network. Redis is used as the rapid decision-making engine for which content to display based on the subscriber demographic.

Reporting and analytics: A healthcare learning management provider uses Redis for reporting and analytics, where, by using Hashes and Sorted sets, they are able to achieve sub-second sorted user responses in analytic scenarios and sub 5 millisecond paging times.

There are of course, hundreds of thousands of Redis users worldwide and a practically unlimited variety of Redis use cases. The above examples provide a glimpse into the horizontal cross-industry nature of Redis use cases.

What does the future look like for Redis?

As big data tools mature into the next generation, we see an increasing focus on real time processing of data for automated decision-making.

As Apache Spark starts to take hold as the general purpose real-time processing framework, Redis, used as an in-memory, shared, distributed data store offers a tremendous boost in performance (45 times higher compared to the performance of Spark using process memory and 100 times higher compared to the performance of Spark using HDFS).

As businesses push towards analytics being done at the speed of business, two things will drive further adoption of Redis:

- An increasing need for a versatile in-memory database that can handle a variety of operational and analytic scenarios at blazing fast performance.
- An increasing availability of cheap memory options such as IBM CAPI Flash, Samsung's NVMe or Intel 3DXpoint which deliver orders of magnitude better performance than Flash for only incremental additional cost.

With further extensions on the horizon for Redis data structures, surging emphasis on a wider variety of processing and analytic scenarios, and Redis Labs providing much of the enterprise grade scalability, high availability and global distribution, Redis is well positioned to lead the in-memory databases as well as the NoSQL market in the coming years.

Summarizing the Redis Advantages

By now, it should be obvious that Redis is all: a database, a cache and also a message broker. It should also be a little clearer that data structures in Redis are similar to some data models in other NoSQLs except they are implemented in memory with the rather unique results of high performance, low computation complexity (for simple and complex operations), high memory space efficiency and low application network traffic. Data structures offer matchless flexibility that make Redis an extremely popular implementation for a variety of use cases.

Trends point to the world needing its data processed faster and faster, and Redis is unparalleled in this regard. As the business world increasingly demands its analytics at the same speed as operations, in-memory processing, storage and analysis of critical data is going to be a massive competitive advantage for Redis. Memory technologies are already shifting to make this a reality from a cost point of view. With Redis set for greater and grander use, judicious usage of the right technologies will buoy the operational capabilities of organizations that choose to deploy them.

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