Editor: **Robert Blumen**
SalesForce
robert@robertblumen.com

Tug Grall on Redis

Akshay Manchale

From the Editor

Tug Grall, technical marketing manager at Redis Labs, discusses topics including Redis basics, data structures, use and management, probabilistic data structures, concurrency model, emerging use cases, RedisGears, scaling up, operational complexity, support for publish-and-subscribe, modules and extensions, and future developments with host Akshay Manchale. To hear the full interview, which was published in January 2021, visit http://www.se-radio.net or access our archives via RSS at http://feeds.feedburner.com/se-radio.—*Robert Blumen*

Akshay Manchale: What is Redis and what is it used for?

Tug Grall: Redis is a not SQL (NoSQL) database for use cases requiring speed, scalability, and flexibility. It began as an open source project that provided a fast dictionary to access data. The user community has added additional features. Redis is well suited to applications that require fast access to data. It is used as a cache, as a session store.

Redis is a simple key-value store in memory to allow application builders to quickly access data multiple times in a session. It can store an e-commerce cart with information about what the user wants to buy and provide fast access to recommendations. Because it's used to share data between different calls or services, it can build a cached catalog for a user session.

> Redis is well suited to applications that require fast access to data.

Redis keeps data resident in memory as opposed to on disk and in databases, which enables operations to complete in 1 ms or less. You configure Redis to also keep data on disk if you have to stop Redis or if you crash. When you restart, you can reload the data from disk to memory, but when you work in your application, the data set is always in memory, which provides predictably fast performance.

Is it a durable messaging system?

It is durable, with the tradeoff of being in memory. It does not reside only in memory; it could be saved on disk. You choose when you want to save. When you configure or start a Redis instance, you can say you want only memory and don't want to store anything on disk, or you can save on disk every 6 hours, 4 hours, and so on. This could be good if you have a big cache, you're exporting a big

Digital Object Identifier 10.1109/MS.2021.3073167
Date of current version: 18 June 2021

data set daily from the mainframe into Redis for fast access, and you don't want all of the processing, even if something crashes in a container. You can also do app unloading—every transaction, every mutation, or every second, you can save data on disks. If you have a persistent log, when you do an *add*, each add is a command, and you can choose to save on disk every time. Everything that is related to applications and to saving on disk is in a different thread to avoid affecting performance.

So Redis is used as a cache on top of other data stores, and you can have updates going into your primary data store to Redis or out of Redis. How do you manage recency of data?

In many ways. You can and should use Redis as your database if the data more than fit and you don't need to present it anywhere else. When used as a cache, the biggest challenge is, when do you invalidate? How do you know that you have to go back to the initial data? The developer has to understand the data they're manipulating. In the simplest case, you set a value for updating, for example, using a web service, and cache for a specified period of time. In this case, as a developer you set the value, call the web service, and it is done for you. When you have something that is more complex, it depends on whether you have access. Can you have some notifications about the initial data source? If you are using a messaging approach, you will use that to invalidate your cache. You can receive the event and say, "When I receive this type of event, I will invalidate the cache." You can do this by deleting the object or asking the system to go into the main data source and put that in the cache.

SOFTWARE ENGINEERING RADIO

Visit www.se-radio.net to listen to these and other insightful hour-long podcasts.

RECENT EPISODES

- 456—Tomer Shiran, cofounder of Dremio, talks about managing data inside a data lake and the ecosystem of products available for storage and analytics with host Akshay Manchale.
- 455—Jamie Riedesel, author of *Software Telemetry Book*, discusses software telemetry, why telemetry data is so important, and the discipline of tracing, logging, and monitoring infrastructure with host Gavin Henry.
- 453—Host Justin Beyer and Aaron Rinehard, chief technology officer of Verica and author, discuss security chaos engineering and how it can be used to enhance security in modern application architectures.

UPCOMING EPISODES

- Host Robert Blumen talks with Jefferey Smith about DevOps antipatterns.
- Daniel Roth discusses Blazor with host Priyanka Raghavan.
- Michael Ashburne and Maxwell Huffman talk with host Jeremy Jung about quality assurance.

Redis is fast and easy to use, and the application programming interface is simple for developers.

Lately, we are using a module called Redis Bloom Filter, part of the ecosystem that is available in open source. Your application can interact only with Redis, but when you modify something in Redis, you will push an update in your relational database. When you work as a developer or architect, you have to first understand how long the data will be valid, what the event is that makes these data invalid, whether this is something you can control, and what the best solution for controlling it will be.

What are some emerging nontraditional uses of Redis?

Redis is fast and easy to use, and the application programming interface is simple for developers. I use hashes to store all my connected users. I have hundreds of thousands of entries, but I would like to do a simple query, such as, "Can you give me the top five connected users coming from California?" I will use a hash to store the user profile that contains the state, but I have to somewhere have a list of

ABOUT THE AUTHOR

AKSHAY MANCHALE is a senior software engineer at Salesforce, where he works on distributed data management systems. He received his master's degree from the University of California Irvine and is passionate about distributed systems, storage, database internals, and systems programming. Further information about him can be found at https://twitter.com/akshayms. Contact him at akshay.manchale@gmail.com.

the states of the connected users to be able to get this value. So I manage my index and the relation between my hash and my list of connected users for a specific state. This is something only advanced developers can use, so the Redis community and Redis Labs developed modules that meet this type of need. One of the first modules was to do an index inquiry on full type search inside Redis.

> Another emerging use case is to be able to query by value in a simple way.

Another emerging use case is to be able to query by value in a simple way. This is done using the Redis search module, which provides more advanced capabilities in querying and aggregation. You don't need to go to a third-party tool to do that. Using the same approach, developers want to create complex applications using graph data and graph databases. They want to be able to quickly identify anomalies or make comparisons to find some pattern and be able to analyze the impact of something. The goal of the community of Redis and the modules was to provide modules that help developers do that.

Where is Redis going in the next couple of years?

Because of the need for a fast, reliable real-time application, Redis is used increasingly as a main database for applications. For this, you need a multiprogramming model with the modules. It's really about adding more commands to give you more features to build a richer application where only Redis is needed not only as a cache but working more as a database.

IEEE Software (ISSN 0740-7459) is published bimonthly by the IEEE Computer Society. IEEE headquarters: Three Park Ave., 17th Floor, New York, NY 10016-5997. IEEE Computer Society Publications Office: 10662 Los Vaqueros Cir., Los Alamitos, CA 90720; +1 714 821 8380; fax +1 714 821 4010. IEEE Computer Society headquarters: 2001 L St., Ste. 700, Washington, DC 20036. Subscribe to *IEEE Software* by visiting www.computer.org/software.

Postmaster: Send undelivered copies and address changes to *IEEE Software*, Membership Processing Dept., IEEE Service Center, 445 Hoes Lane, Piscataway, NJ 08854-4141. Periodicals Postage Paid at New York, NY, and at additional mailing offices. Canadian GST #125634188. Canada Post Publications Mail Agreement Number 40013885. Return undeliverable Canadian addresses to PO Box 122, Niagara Falls, ON L2E 6S8, Canada. Printed in the USA.