Redis Cache – Redis As Primary Database

There are lot of talk on Redis. Redis provides a memory caching technique and improves your application query performance significantly. Today I will summarize few key points on the Redis topic.

When you come for Redis and want to implement it – you probably go through several Redis tutorials, discussion on youTube, google etc. Wherever you go, it will say Redis provides fastest caching service for your data. As if once you store your data to Redis cache, things will go fast.

Many of you must have known it, but for the others let me tell you it is not that straight forward.

Redis Cache is just a Memory Caching Service and Isolated. It has nothing to do with your Primary Database. So given a scenario:

1: You are designing an App/Service for faster Query response.

In a hypothetical way, I presume you would have SQL Server as Primary Database. Then you end-up writing high performed SELECT queries in SQL stored proc by creating some Indexes. But the fact is you may come up with need of making it more performer by implementing Cache and selecting Redis Cache.

With the given scenario now you have 2 Jobs to perform:

1: Maintain high performed query in SQL Server as it is (As Primary Data Source)

2: Maintain/sync the data also in Redis Cache

Then return the data from Redis Cache

When you read many articles, videos …. Hardly people talk about the Step:2 above as pretend as if Step:2 does the entire thing when the reality is, it is you have to maintain the data sync between your Primary database (in this case SQL Database) and Redis Cache. So what you do is - You read the data from SQL Db then store it to Redis Cache yourself. Now at this stage you know - nothing is automatic, there is a new task to load data in Redis Cache and its you have to explicitly maintain the Consistency/Accuracy of data replication between SQL DB and Redis Cache.

For this there are several approaches/patterns available and most precisely they are known as

1. Write-Through Pattern
2. Read-Through Pattern
3. Expiry Pattern

[There are many sub variants also available]

1: **With Write-through**: You use Redis as *Source Of Truth* and then allow Redis to save the changes in SQL DB – Or Primary Database – and the saving can be done asynchronously or in Bulk

2: **Read-Through Pattern**: In this technique, you use ‘Cache Miss’ technique and when no data found in cache, you can read the Primary DB and load the same data in Cache to return response.

3: **Expiry Pattern:** It is applied when there is Command behavior like Create/Update/Delete, you delete the Cache and updates the primary database. So next time users will get data by Cache-Miss technique to be latest.

**The challenges will be now which one above is the best suited for you?**

Because all the above has some Advantages and Minor to Major disadvantages as well.

It is very important that you maintain “Eventual Consistency” in Redis Cache. The industry de facto standard is ‘Write-Through’ Pattern by which you use Redis cache as Source Of Truth and Let Redis update your underlying Primary DB silently in the background.

**Now here is 1 problem.** The technique does not work well in Windows O/S because of Fork Write mechanism that is used by Redis – does not work very well in Windows and There is no official support from Microsoft if you use Redis on Windows ☹

Windows uses WSL2 (Windows Subsystem for Linux -2) to run Redis on Windows and we do not know who is better among ‘Redis on Linux’ vs ‘Redis on Windows’.

Now at this stage, you are landed up with ‘Azure Cache for Redis’ – wait wait

Azure Cache For Redis? Oh yes , this is Azure Managed Service for Redis using ‘Open Source Redis’ remember. You do not have Redis Enterprise Premium Feature here to note…. Any Third Party Redis providers be it Azure, AWS etc using Open Source Redis. Now you no longer need to think Windows issue as it is a fully Azure Managed Service. What you need to do is - Setup Azure Redis Cache and use it, remember the same 2 step process as I said in the very beginning.

*You are Sole responsible for Maintaining data consistency*