Question &Answers

Quest 1: Why did you choose that language?

Answer: I have chosen python language over the other languages because:

- It has a highly scalable architecture with a fully cloud-enabled tech stack from the get-go.
- Flask is a minimalistic framework and does just what you want it to do.
- Asynchronous Redis client.
- Python VM automatically enhances the memory allocated to it , up to the extent the memory for the process is allowable by the OS, if instance starts requiring more resource.

Quest 2: What are the further improvements that can be made to make it efficient?

Answer: There are many scopes of improvements, such as-

- **Multi- processing**-We can use multiprocessing power of python to drastically read and write operations. Redis is single-threaded but it has concurrent I/Os .
- **Better hashing**-For example if we have question_id to usr_id mapping in Redis and we simply store them in a map, it will consume a lot of memory. To conquer this we can use a simple hash type. We bucket all our question_ids into buckets of X. We just take the ids and divide it by X and discard the remainder. That determines which key we fall into; next within the hash, that is stored in the key, the question is the lookup key within the hash and User_id is the value
- **Better Pipelining**- Request/Response server can be implemented so that it is able to process new request even if the client did not already read the old responses. This way it is possible to send ***multiple commands*** to the server without waiting for the replies at all, and finally read the replies in a single step

Quest 3: What data structures have you used and why?

Answer: The data structures I have used are:

- **Strings**: strings are implemented using a C dynamic string library so that we do not have to pay for allocations in append operations
- **Lists**: Used the list data structure for stack using lpush,lpop.
- **Sets**
- **Hashes**: Hashes are the perfect data structure to represent objects, composed f fields and values. Fields of hashes can also be automatically incremented using HINCRBY.Hashes are encoded very efficiently by Redis, and you can ask Redis to automatically GET,SET or Increment individual fields in a very fast manner.
- **Sorted Sets**: Sorted sets are like regular sets that can be used to describe relations but they also allow you to paginate the list of items and to remember the ordering. Sorted sets are good for priority queues.

Quest 4: Does your implementation supports muti-threaded operation? No why can't it be? If yes then how?

Answer: My implementation does not support multi threading because Redis is a single threaded and scales indefinitely in terms of concurrency. A single-threaded program can definitely provide concurrency at the I/O level by using an I/O de/multiplexing mechanism and an event loop (which is what Redis does).

But we can make Redis multi-threaded in some way for performance benefit, the simplest way to think of is that once Redis needs to perform any Write or Read operation, that work is performed by N, previously fanned-out, I/O threads. There's no major complexity added since the main thread is still responsible for the major tasks, but most of the advantages, because the majority of time Redis spends is in IO, as usually, **Redis workloads are either memory or network bound**. Thus adding multi threading/concurrent requests we can drastically improve read and write operations.