Redis Server

It is an in-memory database that allows us to do fast lookups and retrieval. It uses RESP message format for serialization and deserialization.

Data Flow our project:

A diagram of a server

AI-generated content may be incorrect.

For speed, the Redis server is implemented using sockets and uses TCP connection as RESP is more suitable with TCP connections.

Input supports the following command:

1. String Commands
   1. SET <key> <value> - set the string value of a key
   2. GET <key> - get the value of a key
   3. DEL <key> - delete a key
   4. EXPIRE <key> <seconds> - set a timeout on the key
2. Hash Commands
   1. HSET <key> <field> <value> - set the field – value pair with the hash specified in the key.
   2. HGET <key> <field> - get the value of a field with hash specified in the k
   3. HGETALL key – get all the field values with the hash specified in the key.
3. Integer Commands
   1. INCR <key> - increment the integer value of the key by one
   2. DECR <key> - decrement the integer value of the key by one
   3. INCRBY – increment the integer value of the key by specified amount
   4. DECRBY – decrement the integer value of the key by specified amount
4. List Commands
   1. LPUSH <key> <value> - insert the value at the head of the list.
   2. LPOP <key> - Remove and return the first element of the list.
   3. LRANGE <key> <start> <stop> - get a range of elements from the list.
5. Set (Data Structure) Commands
   1. SADD <key> <value> - add a member to the set
   2. SREM <key> <value> - remove a member from the set.
   3. SMEMBERS <key> - get all the members of a set.
6. Sorted Set Commands
   1. ZADD <key> <score> <member> - add a member to the sorted set
   2. ZRANGE <key> <start> <stop> - get a range of elements from the sorted set
   3. ZREM <key> <member> - remove am member from the sorted set.
7. Transaction Commands
   1. MULTI – start a transaction
   2. EXEC – execute all commands after MULTI
   3. DISCARD – discard all commands issued after MULTI

Output from the Redis Server supports the following RESP style data types:

1. Simple Strings – first byte represented by +.
2. Errors – first byte of reply is -.
3. Integers – first byte of reply is :
4. Bulk Strings – first byte of reply is $
5. Arrays – first byte of reply is \*
6. Null Bulk strings - first byte of reply is $-1
7. Null Arrays - first byte of reply is \*-1

Persistence mechanisms we have used in the project:

1. AOF – we append all the write operations into a single file and replay this file on Redis server restart or crash.

Other features implemented:

1. Pipelining user commands so they can send multiple commands and wait for replies later.
2. The server changes to push protocol upon user subscribing to pub/sub channel.
3. Monitoring feature, we can check for changes.