HW 2B REPORT

TASK-2

Name	UIN
Venkata Koustubha Krishna Pydimarri	526005194
Ravi Teja Muppala	526003004

Time Tracking :

Total Time : 5 hrs

No.	Day	Time	Duration/hr	Tasks
1	10/29/2018	10:30am-11:00am	15min	Understanding the requirements of the project
2	10/29/2018	11:00am-11:30am	30min	Going through Redis tutorial videos and basic commands
3	10/29/2018	11:30pm-12:15pm	15min	Installing Redis and setting up the environment for the task
4	10/29/2018	1:00pm-3:30pm	2hr 30min	Implementing the message board architecture using Python
5	10/29/2018	3:30pm-5:00pm	1hr 30min	Writing the report for Task-2

System Architecture

Database choice

The primary goal of this task is to have a blueprint for the message board application. The databases that we are trying to use here are MongoDB and Redis. Both are NoSQL databases. NoSQL databases are chosen because they offer better advantages compared to traditional relational databases.

Redis is used for the channel listening for the design of message board system as it is an in-memory database and generally faster than MongoDB. Redis offers the benefits of low latency while receiving updates for its users through its PUB/SUB methods. As the data meant for listening is not that much, it's reasonable to put them in memory. Hence, Redis is a good choice for listening to the updates.
MongoDB is the optimal choice for data storage as it is more scalable, persistent and can hold more data. MongoDB’s scalability feature helps us to add more messages for storage from time to time.

Design

The design of system is shown in Fig. 1. The clients write messages both to Redis (locally) and MongoDB (through Internet). While clients read, a collection of JSON objects will be retrieved from MongoDB, then parsed into an array of messages. In order to implement the listen function, Redis SUB/PUB method is used. Clients will keep monitor the specified channel of Redis while listening. When new writes happen, they will be notified.

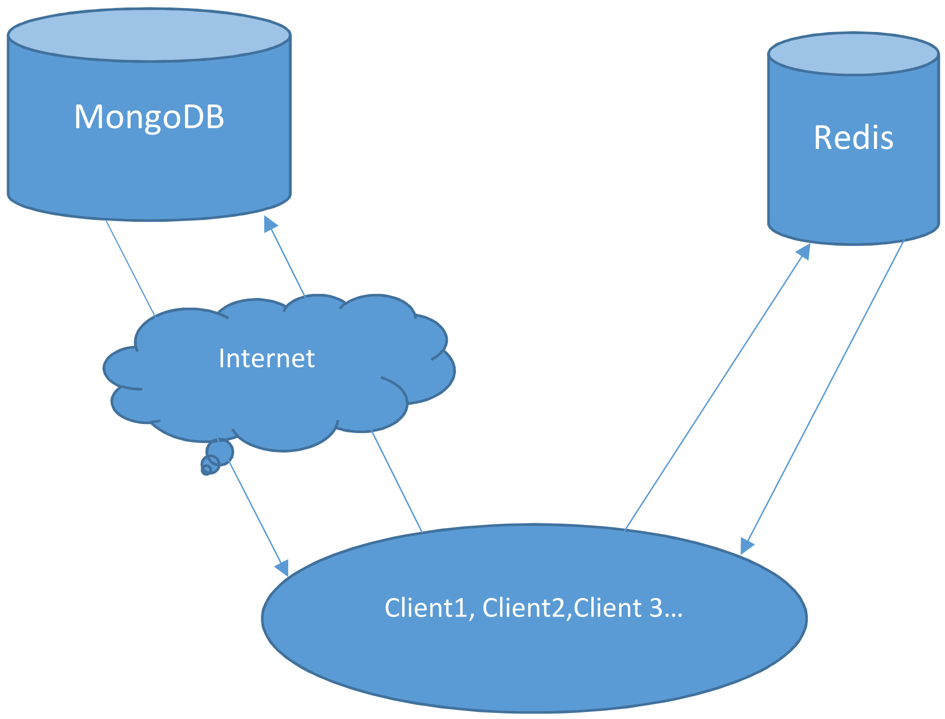


Fig-1: System Architecture

Prototype

The Redis commands that we are going to use for this task and the role played by MongoDB is explained below. The Redis server is first started so that the channels can perform necessary operations for the message board application. The program is basically an infinite while loop receiving commands listed below.

Operation	Purpose
SELECT	This command is used for starting and enabling a particular channel. This is the first command that should be executed before any other operation. Otherwise, an error is displayed indicating the same.
READ	This command enables the channel to read all messages from the selected message board. When no message board is selected, it results in an error
WRITE	This command enables you to write the message into the selected message board. When no message board is selected, it results in an error.
LISTEN	When this command is used, it waits to get real time updates on the selected message board. When no message board is selected, it results in an error.
STOP	This command enables one to stop the user from listening to messages and after issuing this command, the possibility of performing necessary operations is eliminated.
QUIT	Just terminates the execution

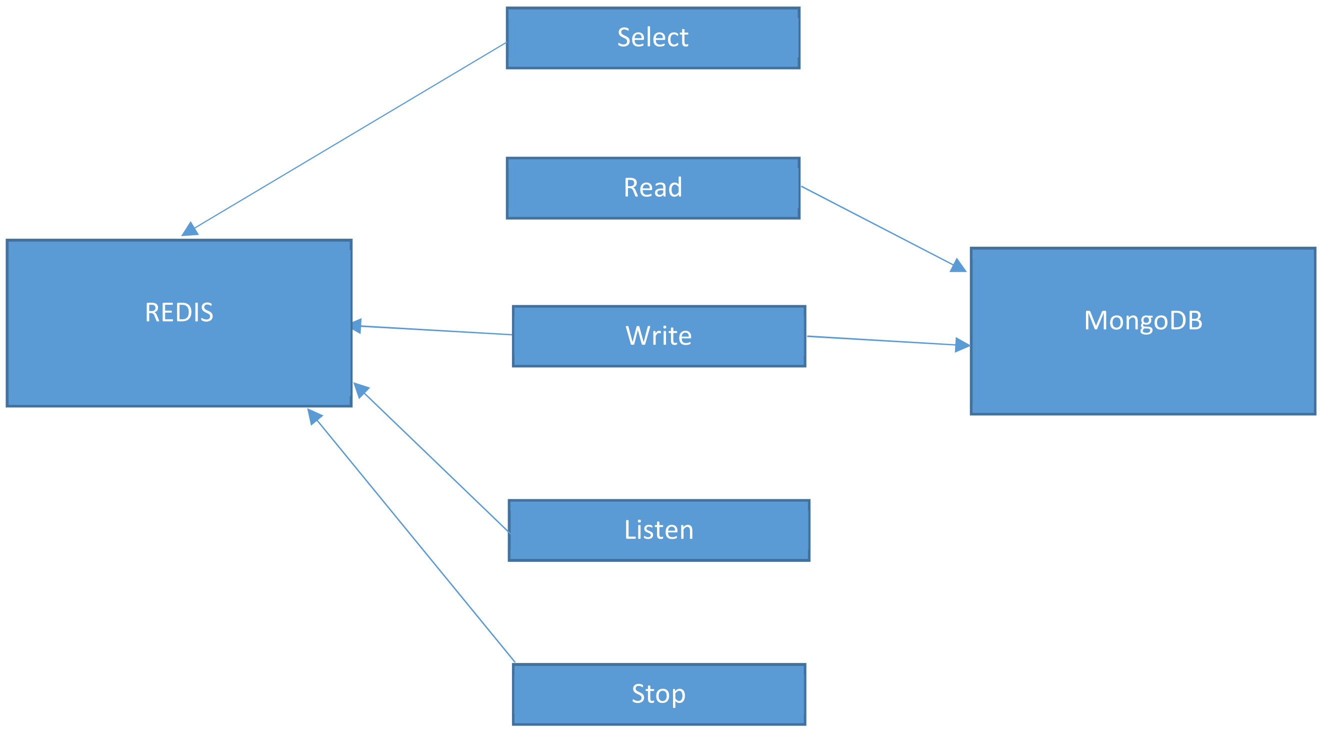
Compile and run

1. Make sure Redis is installed and running locally.
2. Run **python control.py** provided in our repository to create a client. You can repeat this in different terminals as many as you want to create many clients.
3. The MongoDB database used is the remote one provided as described in **constants.py**

4. The **MongoConnect.py** enables users to connect to MongoDB database.
5. To run Redis Server, just use **redis-server** command.
6. To shutdown Redis server, **redis-cli shutdown** is the command.

Link for our Github Repository--- https://github.tamu.edu/Ravi/HW2

Diagram illustrating the operations that are carried out on each database :


Select
Read
REDIS
Write
MongoDB
Listen
Stop