Programming Assignment 6

Shubham Adep

Experiments Performed

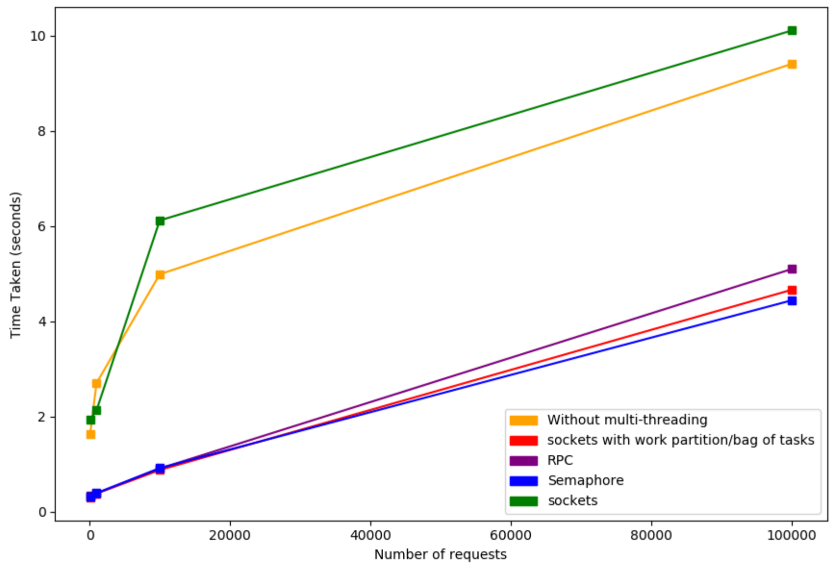
Implementing a dictionary in go using:

1. Basic implementation without multi-threading and sockets
2. Sockets
3. Sockets with bag of tasks/work partition
4. RPC
5. Semaphore

In this report, the above 5 approaches have been referred in the order specified above.

Which system is scalable ?

The graph below shows the performance of all the approaches with respect to time when the number of read requests are increased. Sockets without go routines performed the worst, even worse than a simple implementation of the dictionary without sockets due to communication overhead. Approach 3 with bag of tasks and work partition have similar performances and hence a common line is considered in the graph. RPC performance worse than sockets with go routines because they have an extra over-head and are used for ease of deployment, whereas sockets are low level, and hence sockets with go routines perform better and comparable with last experiments with Semaphore as in the case of comparison only read requests are considered which makes both the implementations the same.



Which system has less code complexity than others ?

Below table shows the number of lines used for each approach. In the table it is evident that even though go is such a distributed systems “friendly” language, including features like semaphore and write requests can increase the complexity. Also, as RPC is easier to implement than basic sockets, you can observe the reduction of lines of code from 128 to 104.

Implementation of dictionary	Lines of Code
without sockets	54
with sockets	119
sockets with bag of tasks/work partition	128
RPC	104
Semaphore	225

Which system is robust one ?

Approach which uses Semaphore is more robust and more functional as it allows write/read requests. RPC and sockets are reliable and more robust than the basic implementations with or without sockets as due to multi-threading the system becomes more scalable and hence making it more robust for large number of incoming requests.

Conclusion

We explored various methods to implement a distributed system and analyzed them in terms of complexity, performance and robustness.