Golang Data fetch memory optimization and testing:

Task: -

1. Postgres setup and SQL
2. Fetch Data from Postgres
3. Client to receive data
4. Memory optimization
5. Analysis of memory
6. Postgres setup and SQL: Setting up postgres using Docker and connecting DBeaver (SQL Editor) to insert table and data.
7. Fetch Data from Postgres: Fetching data from postgres database using golang “database/sql” and publishing them using “net/http” package.
8. Client to receive data: Reading the content of a webpage using “net/http” package.
9. Memory optimization: To reduce the constant memory retention of golang “runtime/debug” free up the space used, back to the OS this helps us to clear the RAM.
10. Analysis of memory: PPROF is a golang tool which gives us a profile with which we can see the detailed information.

To see detailed information in

Web: -

* localhost: \_\_\_\_/debug/pprof

Download options: -

* localhost: \_\_\_\_/debug/pprof/profile
* localhost: \_\_\_\_/debug/pprof/heap

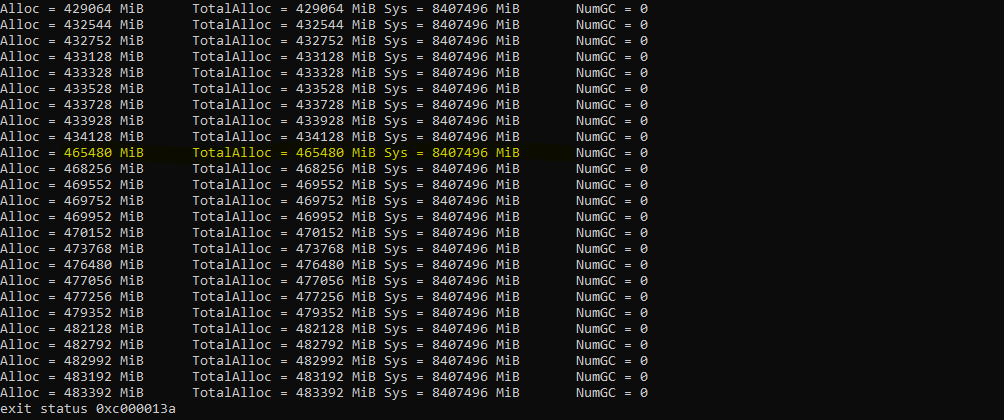
How to use pprof:

Steps:

1. Use the cmd
2. Go tool pprof \_\_\_(profile name)
3. Get report

Analysis of Memory:

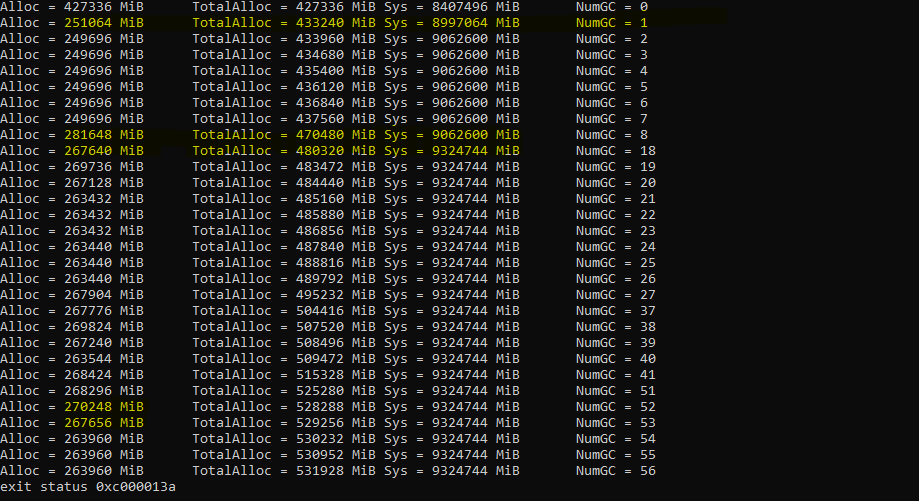
Without debug.FreeOSMemory:



Observation: The memory constantly is increasing and at time of requesting the response the memory increased drastically and it

Keep on increasing. The Garbage collection is also zero.

With debug.FreeOSMemory:



Observation: The memory constantly is increased and after that it decreased and at the time of response it increased due to the store and fetch of data after that the fetch didn’t increase much .

When the DB has 1 billion and fetch data from the DB.

* The golang database/sql is used to access the database and then open the DB using Query we put the pointer and fetch the rows one by one using rows.Next .

Before OptimizationDiagram, schematic

Description automatically generated

After Optimization

Diagram, schematic

Description automatically generated

* Why Postgres?

Postgres is used in industries because it gives the ability to use the database regardless of quantity of data. It also offers a lot of functionality to work.

* Why Golang?

It is better because managing thread in golang is easy using go routines and this property thing made with golang highly scalable and the garbage collection is also useful for memory.

* Ran out of memory error

10 - Successful

100 - Successful

1000 - Successful

10000 - Successful

100000 - Successful

1000000 - Successful

1304257 – Maximum Successful

10000000