‘Go’ faster with data driven Optimizations

Ravi Gudimetla - Software Engineer

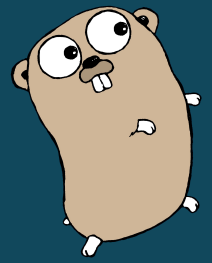
redhat.

What is the talk about?

- Golang... obviously
- Why do we need to benchmark?
- Introduce some basic benchmarking tools
- Analyzing benchmarking data from call graphs

redhat.

Let’s code !!!



- Let us look at simple in-memory KV store with Get and Put operations
  - Existing Implementation
    - Hmm, I think I can do better than that
  - Modified Implementation
    - Looks good
- How do you prove that the second implementation is better than the first?

redhat.

Benchmarking & Profiling to the rescue

- Go testing package helps in automated testing of code
- Functions of form BenchXXX are considered as benchmarks
- These could be run with “go test”
- More interestingly we could profile our code to identify bottlenecks. Yay!!!
- For our code, benchmarks look like this

redhat.

How to benchmark and generate call graph

- To run benchmarks and generate a binary and profile information:

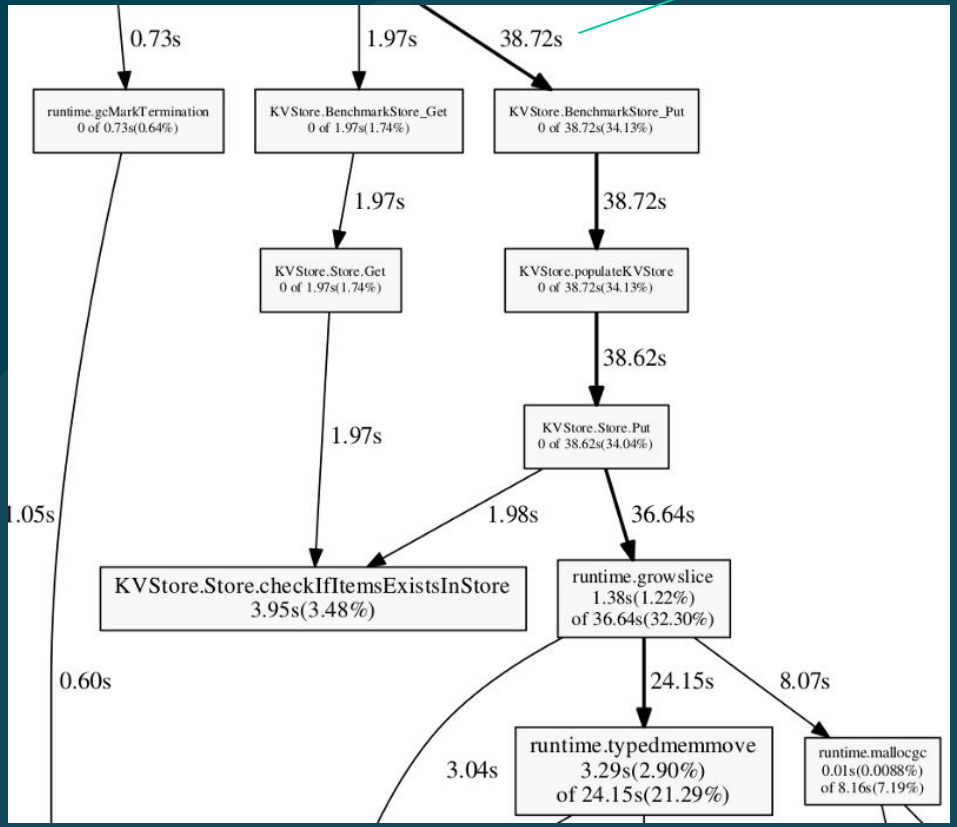
```
go test -run=^$ -bench=. -cpuprofile=cpu.out
```

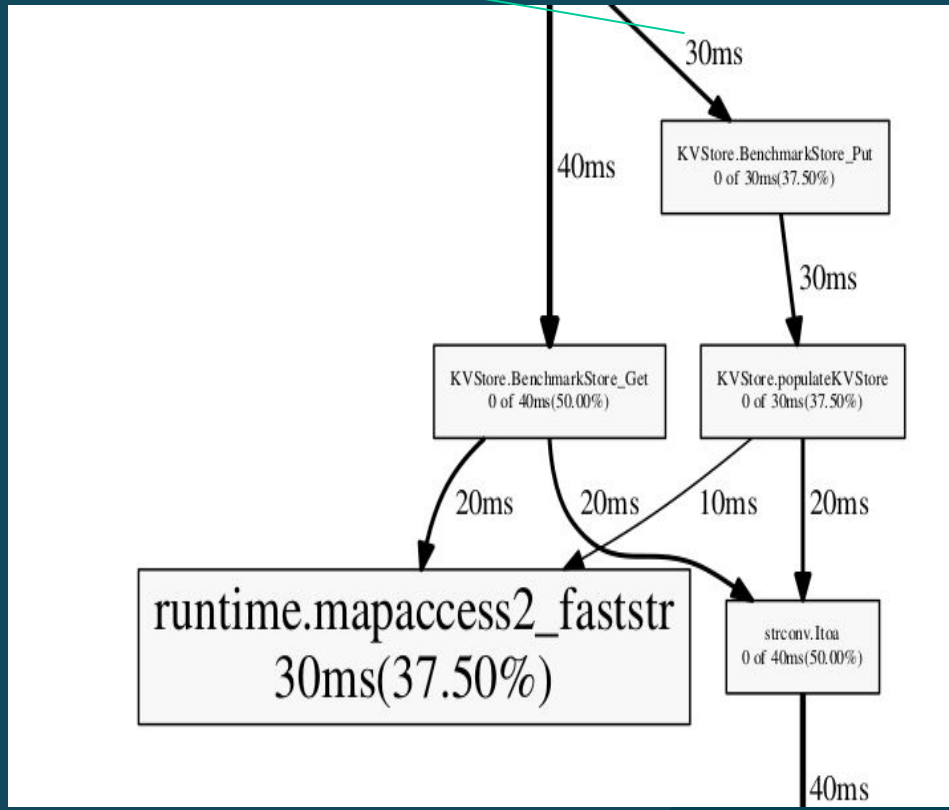
- To generate a call graph:

```
go tool pprof KVStore.test cpu.out
Entering interactive mode (type “help” for commands)
(pprof) web
```

redhat.

> 1000x improvement





Recap

- Why do we need to benchmark & profile?
- How to benchmark?
- How to analyze call graphs?

Warning:

“Lies, Damned Lies and Benchmarks” - Unknown

Links:

Blog post

https://github.com/ravisantoshgudimetla

redhat.

Interested in this kind of work?

We're hiring

https://careers-redhat.icims.com/jobs/search?ss=1&searchLocation=12781-12805-Boston

300 A Street, Boston MA

redhat.

Profiling in Golang

-

redhat.