pprof for Beginners

Miriah Peterson

Intro

- Programs use memory and cpu resources
- Microservices/cloud deployments have conditioned developers ignore how software interacts with its environment
- Understanding how your program uses its memory and cpu can help developers understand some weak points of their services.

Service Weak Points

- What is a memory leak?
- What is a database collision?
- What is a mutex loc?
- What is the job of the garbage collector?
- Should I share memory across my goroutines?
- Should this procedure be inlined?
- How is the compiler optimizing my code?

What is Profiling

> In software engineering, profiling ("program profiling", "software profiling") is a form of dynamic program analysis that measures, for example, the space (memory) or time complexity of a program, the usage of particular instructions, or the frequency and duration of function calls.

Wikipedia - Profiling(computer programming)

What is pprof

What information can pprof provide?

P99 CONF

What does pprof do - allocations

“ *allocs*: A sampling of all past memory allocations ”

What does pprof do - memory

> “ *heap*: A sampling of memory allocations of live objects. You can specify the gc GET parameter to run GC before taking the heap sample. ”

What does pprof do - cpu

> *profile*: CPU profile. You can specify the duration in the seconds GET parameter. After you get the profile file, use the go tool pprof command to investigate the profile.

What does pprof do - goroutine

“ *goroutine*: Stack traces of all current goroutines. Use debug=2 as a query parameter to export in the same format as an unrecovered panic. ”

What does pprof do - others

- *block*: Stack traces that led to blocking on synchronization primitives
- *cmdline*: The command line invocation of the current program
- *mutex*: Stack traces of holders of contended mutexes
- *threadcreate*: Stack traces that led to the creation of new OS threads
- *trace*: A trace of execution of the current program. You can specify the duration in the seconds GET parameter. After you get the trace file, use the go tool trace command to investigate the trace.

Setup pprof - server

```
import (
        _ "net/http/pprof"
)
func main() {
        // run pprof
        go func() {
                fmt.Println(http.ListenAndServe("localhost:6060", nil))
        }()
    ...
}
```

Setup pprof - files

```
if err := pprof.StartCPUProfile(f); err != nil {
        log.Fatal("could not start CPU profile: ", err)
}

// remove unused heap allocations

runtime.GC()
if err := pprof.WriteHeapProfile(f); err != nil {
        log.Fatal("could not write memory profile: ", err)
}
```

Setup pprof - chi router

```
import (
        "github.com/go-chi/chi"
        "github.com/go-chi/chi/middleware"
)

func main() {
        r := chi.NewRouter()
        r.Mount("/debug", middleware.Profiler())
}
```

chi

Using pprof - default

After adding pprof driver to your go server, default web page is available via server for consuming these endpoints.

- code for this example is available here
- demo

Using pprof - cli

To access and analyze pprof data, you can use the cli command:

```
go tool pprof http://localhost:6060/debug/pprof/allocs
```

demo

additional docs

Using pprof - UI

You can also access and analyze pprof data via a helpful web UI.
If you want to access the interactive web UI you will need to install the graphviz tool

```
go tool pprof -http=:18081 http://localhost:6060/debug/pprof/profile\?seconds\=30
```

demo

Using pprof - save files

If you have cached your data, access it directly from the stored protobuf files

```
go tool pprof ex-4-benchmarking/solution/cpu.prof
```

pprof FAQ

- Should I run pprof in production?
- Does it impact my go application?
- How do I learn more about prof?

In Summary

Use pprof as a tool for debugging and improving your code. It will unlock the memory contingent tools like mutexes, goroutines, cpu threading, and go channels.

Contact

- Twitter @captainnobody1
- GitHub soypete
- LinkedIn

REFERENCE:

- https://github.com/google/pprof
- https://perf.wiki.kernel.org/index.php/Main_Page
- https://github.com/google/pprof/blob/main/doc/README.md
- https://en.wikipedia.org/wiki/Profiling_(computer_programming)#:~:text=In software engineering%2C profiling (",and duration of function calls)
- https://go.dev/doc/diagnostics#profiling
- https://www.infoq.com/articles/debugging-go-programs-pprof-trace/

- https://pkg.go.dev/runtime/pprof
- https://pkg.go.dev/net/http/pprof
- https://go.dev/blog/pprof
- https://en.wikipedia.org/wiki/Call_stack#:~:text=In computer science%2C a call,to just "the stack".
- https://en.wikipedia.org/wiki/Point_location
- https://go-chi.io/#/pages/middleware?id=profiler