Profile-guided optimization

Krzysztof Dryś

2023-10-12

Improve performance of you app by up to 7% using this one simple trick

PGO lifecycle

1. `go build .`
2. Run your application in production,
3. `curl -o cpu.pprof "http://localhost:8080/debug/pprof/profile?seconds=30"` to profile your application,
4. `go build -pgo cpu.pprof .`
5. Deploy new profile to prod,
6. Observe performance gains.

Demo

Haversine formula

The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles. from wikipedia

Implemented by github.com/jftuga/geodist.

```
var elPaso = geodist.Coord{Lat: 31.7619, Lon: 106.4850}
var stLouis = geodist.Coord{Lat: 38.6270, Lon: 90.1994}
miles, km = geodist.HaversineDistance(elPaso, stLouis)
fmt.Printf("[Haversine] El Paso to St. Louis:  %.3f m, %.3f km\n", miles, km)
```

Results

```
goos: linux
goarch: amd64
pkg: github.com/krzysztofdrys/pgo-talk/benchmarks/distance_paralel
cpu: Intel(R) Xeon(R) CPU @ 2.80GHz
            | nopgo.tests.times |       pgo.tests.times        |     pgo_v2.tests.times       |
            |       sec/op       |   sec/op     vs base         |   sec/op     vs base         |
Distance-2         92.10n ± 0%   89.97n ± 0%  -2.31% (n=100)   89.94n ± 0%  -2.35% (n=100)
```

JSON marshalling

```
func BenchmarkJson(b *testing.B) {
    b.StopTimer()
    // Reads 1.1M of json data
    cs, err := city.Read()
    if err != nil {
        panic(err)
    }
    b.StartTimer()

    for i := 0; i < b.N; i++ {
     _, err := json.Marshal(cs)
         if err != nil {
          panic(err)
      }
    }
}
```

Results

```
goos: linux
goarch: amd64
pkg: github.com/krzysztofdrys/pgo-talk/benchmarks/json
cpu: Intel(R) Xeon(R) CPU @ 2.80GHz
       │ nopgo.tests.times │       pgo.tests.times        │      pgo_v2.tests.times       │
       │      sec/op       │   sec/op     vs base          │   sec/op     vs base          │
Json-2        4.302m ± 0%   3.748m ± 0%  -12.90% (n=100)   3.816m ± 0%  -11.32% (n=100)
```

Will profiles from version `1.0.1` work for `1.0.2`?

Yes, mostly.

Source stability

Specifically, Go uses line offsets within functions (e.g., call on 5th line of function foo).

Many common changes will not break matching, including:

- Changes in a file outside of a hot function (adding/changing code above or below the function).
- Moving a function to another file in the same package (the compiler ignores source filenames altogether).

Some changes that may break matching:

- Changes within a hot function (may affect line offsets).
- Renaming the function (and/or type for methods) (changes symbol name).
- Moving the function to another package (changes symbol name).

(from go documentation)

What optimisations are