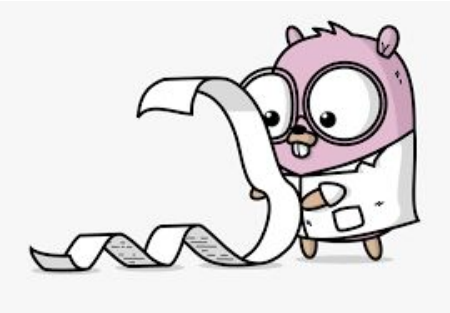
Профилирование и тестирование Go программ



Автор: Денис Лимарев

How to actually learn any new programming concept

Essential

Changing Stuff and Seeing What Happens

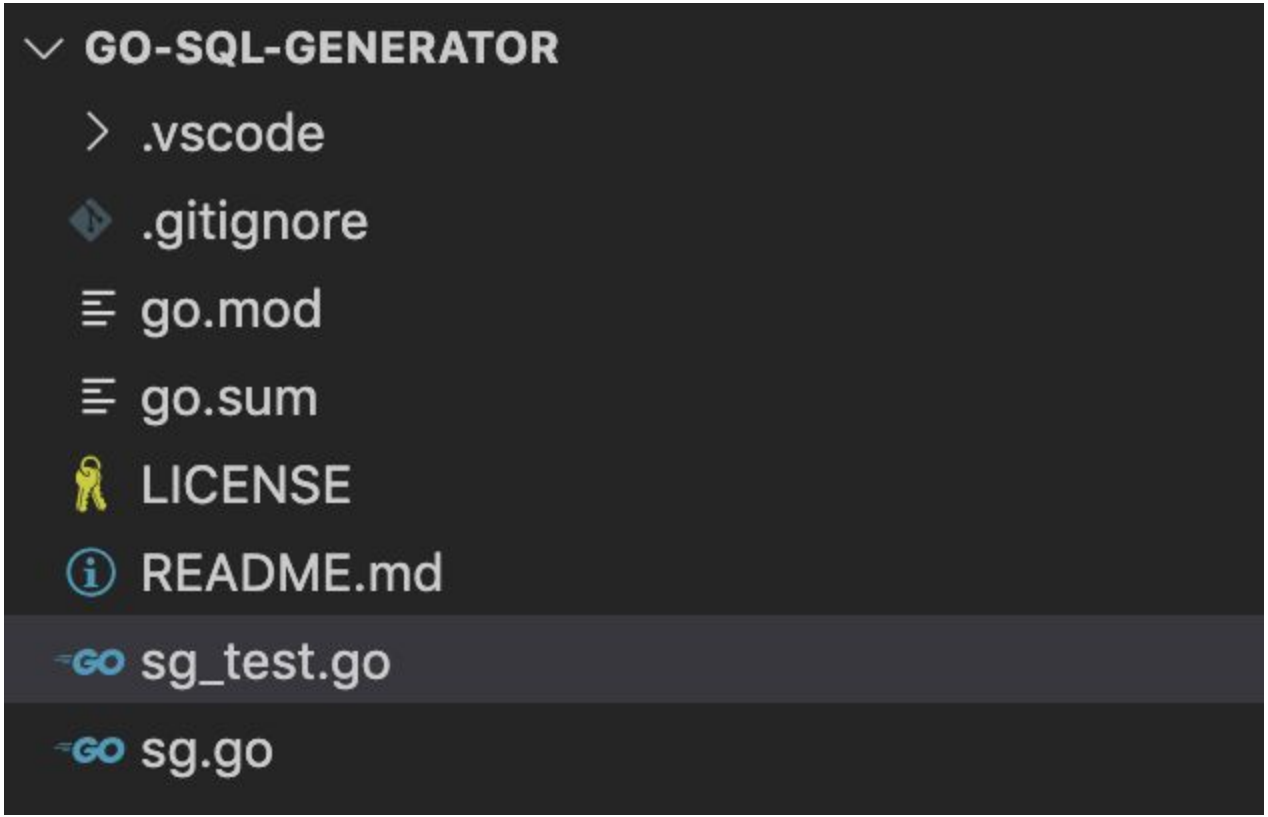
O RLY?

@ThePracticalDev

Тестирование

Плюсы:

- Встроенная библиотека для тестирования
- Возможность параллельного выполнения
- Вывод протестированного кода по блокам и в процентах
- Встроенная поддержка бенчмарков
- Вывод информации по количество операций аллокаций памяти
- Встроенная поддержка моков


GO-SQL-GENERATOR
.vscode
.gitignore
go.mod
go.sum
LICENSE
README.md
sg_test.go
sg.go

```
103
        run test | debug test
104     func TestGetInsertSQLWithOptimize(t *testing.T) {
105         tests := []struct {
106             name         string
107             fields       []string
108             valuesStack []struct {
109                 Values []string
110                 ID     string
111             }
112             expected map[string]string
113         }{
114             {
```

```
162
163         values = append(values, "("+strings.Join(namedParams, ", ")+")")
164
165         namedParams = namedParams[:0]
166     }
167
168     if data.IsIgnore {
169         ignore = "IGNORE"
170     }
171
172     if data.IsOptimize() {
173         sort.Sort(data.ValuesList)
174     }
175
176     var sql = fmt.Sprintf(
177         "INSERT %s INTO %s (%s) VALUES %s",
178         ignore,
179         data.TableName,
180         strings.Join(data.Fields, ", "),
181         strings.Join(values, ", "),
182     )
```

Режимы тестирования

- Тесты
- Бенчмарки
- Параллельные тесты
- Тест на main
- Фазинг (драфт)

```go
    for _, test := range tests {
        valuesCount = 0
        dataInsert = InsertData{
            TableName: "TestTable",
            Fields:    test.fields,
        }

        for _, values := range test.valuesStack {
            dataInsert.Add(values)
            valuesCount += len(values)
        }

        query, args, err := sqlGenerator.GetInsertSQL(dataInsert)
        if err != nil {
            t.Fatalf("on GetInsertSql: %s", err)
        }

        if actualValuesCount := strings.Count(query, "?"); actualValuesCount != valuesCount {
            t.Fatalf("on compare values count and params: expected: %d, actual: %d", valuesCount, actualValuesCount)
        }

        if actualValuesCount := strings.Count(query, "?"); actualValuesCount != len(args) {
            t.Fatalf("on compare values count and args: expected: %d, actual: %d", actualValuesCount, len(args))
        }

        if valuesCount != len(args) {
            t.Fatalf("on compare values count and args: expected: %d, actual: %d", valuesCount, len(args))
        }
    }
```

Профилирование

Плюсы:

- Встроенная библиотека для профилирования
- Встроенная поддержка профилирования хендлеров
- Профилирование по памяти/CPU
- Возможность построения флейм графа или графа выполнения

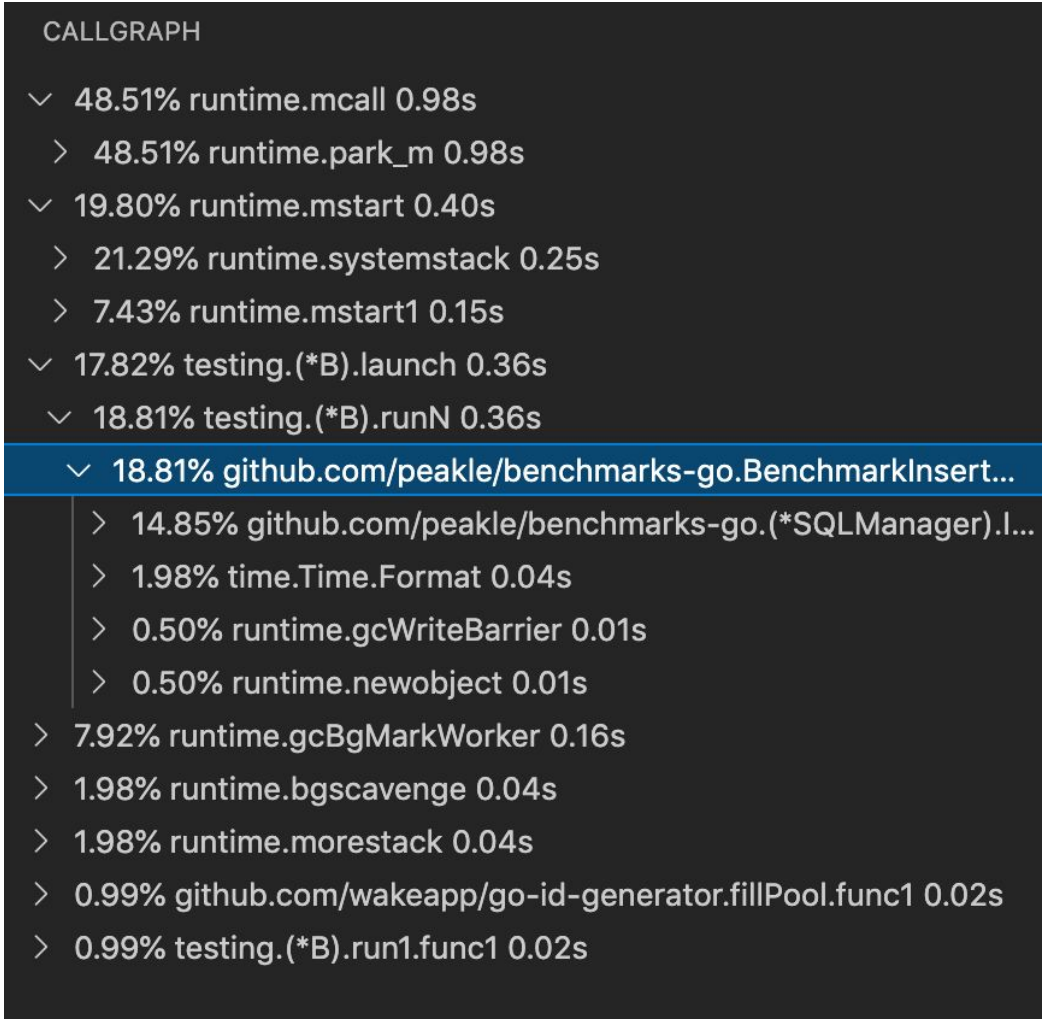
Минусы:

- Необходимость ставить дополнительные библиотеки для построения дерева выполнения (в случае консольных команд)

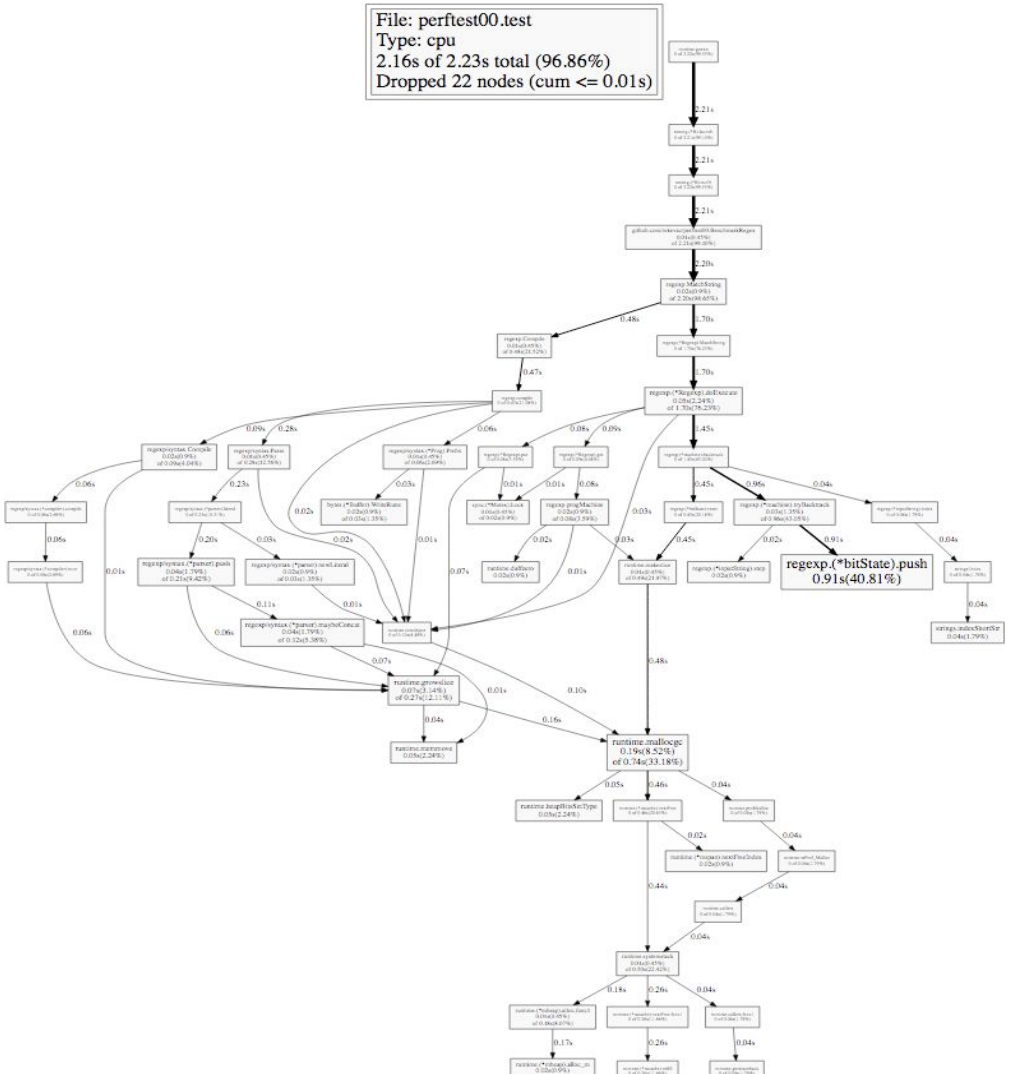
```go
func BenchmarkSubstring(b *testing.B) {
        for i := 0; i < b.N; i++ {
                strings.Contains(haystack, "auctor")
        }
}

func BenchmarkRegex(b *testing.B) {
        for i := 0; i < b.N; i++ {
                regexp.MatchString("auctor", haystack)
        }
}
```

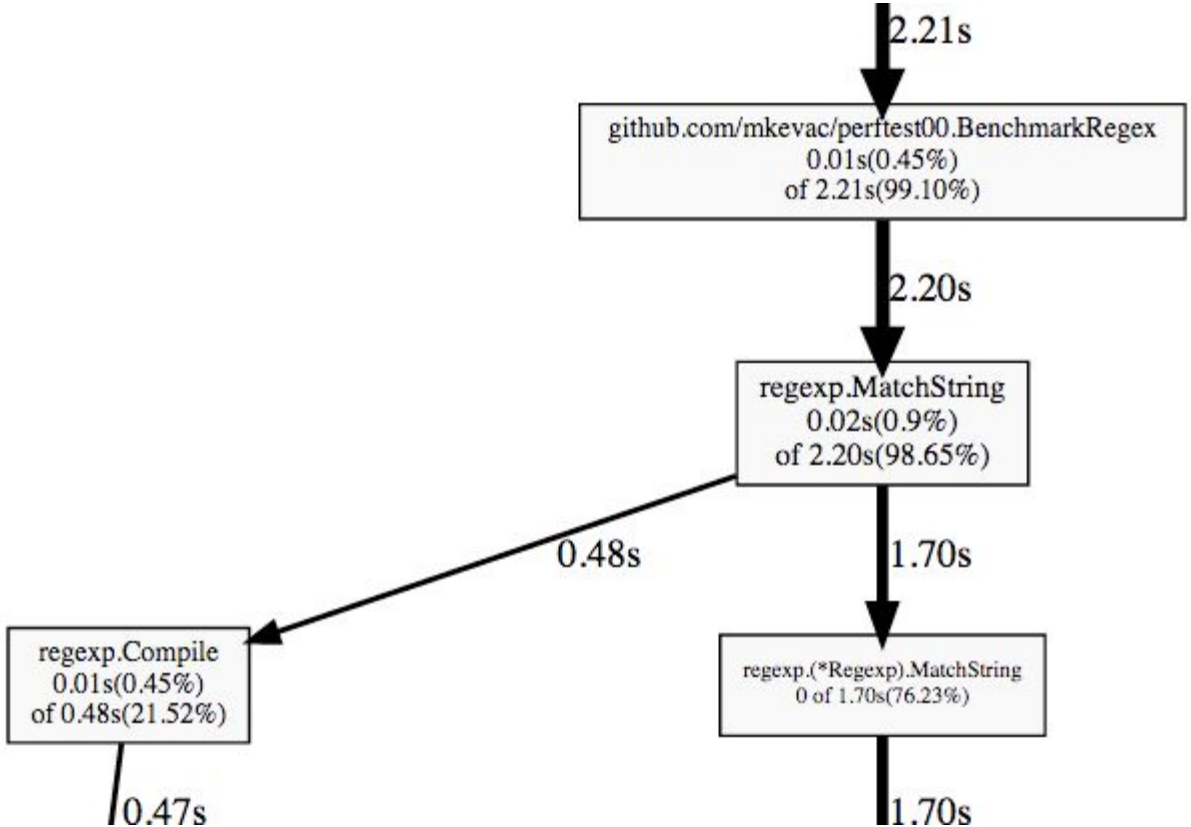
```
$ go test -bench=.
testing: warning: no tests to run
BenchmarkSubstring-8    10000000               194 ns/op
BenchmarkRegex-8          200000              7516 ns/op
PASS
ok      github.com/mkevac/perftest00    3.789s
```

```
$ GOGC=off go test -bench=BenchmarkRegex -cpuprofile cpu.out
testing: warning: no tests to run
BenchmarkRegex-8          200000              6773 ns/op
PASS
ok      github.com/mkevac/perftest00    1.491s
```


CALLGRAPH
48.51% runtime.mcall 0.98s
48.51% runtime.park_m 0.98s
19.80% runtime.mstart 0.40s
21.29% runtime.systemstack 0.25s
7.43% runtime.mstart1 0.15s
17.82% testing.(*B).launch 0.36s
18.81% testing.(*B).runN 0.36s
18.81% github.com/peakle/benchmarks-go.BenchmarkInsert...
14.85% github.com/peakle/benchmarks-go.(*SQLManager).I...
1.98% time.Time.Format 0.04s
0.50% runtime.gcWriteBarrier 0.01s
0.50% runtime.newobject 0.01s
7.92% runtime.gcBgMarkWorker 0.16s
1.98% runtime.bgscavenge 0.04s
1.98% runtime.morestack 0.04s
0.99% github.com/wakeapp/go-id-generator.fillPool.func1 0.02s
0.99% testing.(*B).run1.func1 0.02s

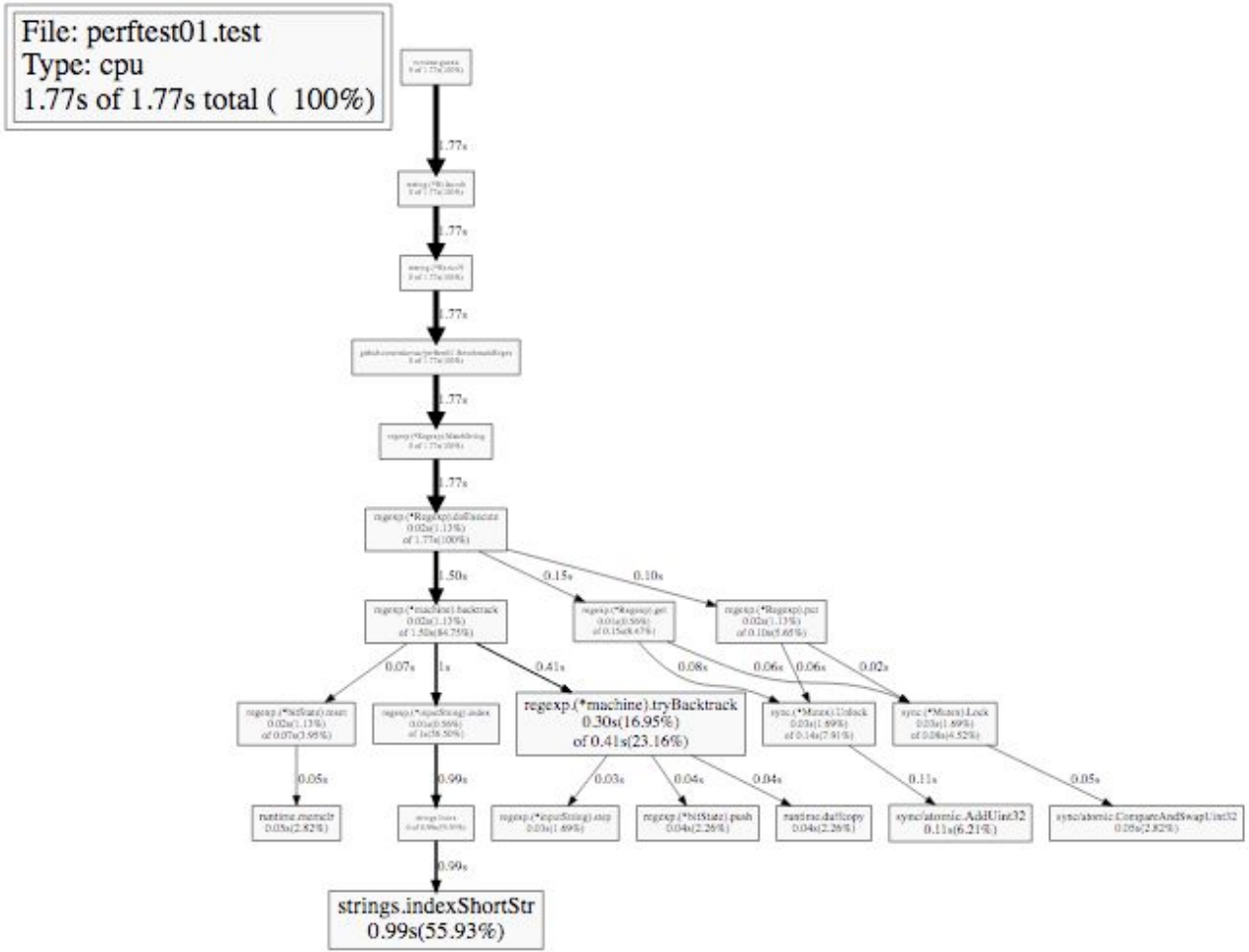
```
$ go tool pprof perftest00.test cpu.out
```

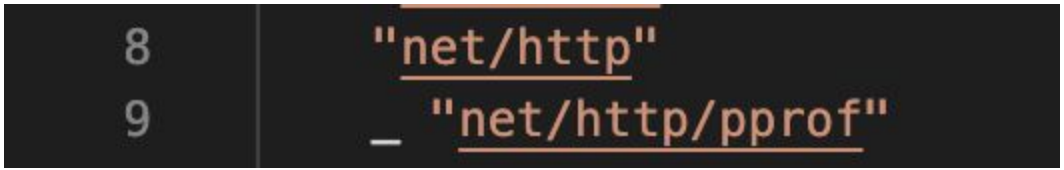

File: perftest00.test
Type: cpu
2.16s of 2.23s total (96.86%)
Dropped 22 nodes (cum <= 0.01s)
regexp.MustString
0.02s(0.9%)
of 2.20s(98.65%)
regexp.Compile
0.01s(0.45%)
of 0.48s(21.52%)
regexp.(*Regexp).doExecute
0.05s(2.24%)
of 1.70s(76.23%)
regexp.(*Regexp).tryBacktrack
0.01s(0.45%)
of 0.46s(20.63%)
regexp.(*bitState).push
0.91s(40.81%)
runtime.growslice
0.07s(3.14%)
of 0.27s(12.11%)
runtime.mallocgc
0.19s(8.52%)
of 0.74s(33.18%)


2.21s
github.com/mkevac/perftest00.BenchmarkRegex
0.01s(0.45%)
of 2.21s(99.10%)
2.20s
regexp.MatchString
0.02s(0.9%)
of 2.20s(98.65%)
0.48s
1.70s
regexp.Compile
0.01s(0.45%)
of 0.48s(21.52%)
regexp.(*Regexp).MatchString
0 of 1.70s(76.23%)
0.47s
1.70s

```
var haystack = `Lorem ipsum dolor sit amet, consectetur adipiscing
[...]
Vivamus vitae nulla posuere, pellentesque quam posuere`
var pattern = regexp.MustCompile("auctor")

func BenchmarkSubstring(b *testing.B) {
        for i := 0; i < b.N; i++ {
                strings.Contains(haystack, "auctor")
        }
}

func BenchmarkRegex(b *testing.B) {
        for i := 0; i < b.N; i++ {
                pattern.MatchString(haystack)
        }
}
```

```
$ go test -bench=.
testing: warning: no tests to run
BenchmarkSubstring-8    10000000                170 ns/op
BenchmarkRegex-8         5000000                297 ns/op
PASS
ok      github.com/mkevac/perftest01    3.685s
```


File: perftest01.test
Type: cpu
1.77s of 1.77s total (100%)
1.77s
1.77s
1.77s
1.77s
1.77s
1.50s
0.15s
0.10s
0.07s
1s
0.41s
0.08s
0.06s
0.06s
0.02s
0.05s
0.99s
0.03s
0.04s
0.04s
0.11s
0.05s
0.99s
regexp.(*machine).tryBacktrack
0.30s(16.95%)
of 0.41s(23.16%)
strings.indexShortStr
0.99s(55.93%)


8 "net/http"
9 _ "net/http/pprof"

Полезные ссылки

1. https://github.com/peakle/benchmark-go [Пример бенчмарков]
2. https://github.com/wakeapp/go-sql-generator [Пример тестов]
3. https://habr.com/ru/company/badoo/blog/301990/ [Статья про профилирование в Go]
4. https://docs.google.com/spreadsheets/d/18giT_NbYLo9yc7yArAeTMnB8W9m3EqUvwftrfZMUyWQ/edit#gid=0 [Полезные материалы по тестированию]

Спасибо за внимание