

GO-JEK



Metric Tracking



and

Nuances of Memory Management

A bit more about me
Gopher @ Gojek
Love to code
Also known as Blackvirus

So what are we trying to do



What is an order?

Interaction between 3 parties Merchant,
Driver, Customer

Created -> Waiting for Driver Allocation -> Driver
OTW to pickup -> Driver Otw to Destination ->
Complete

1. Create
2. *GetState*
3. Allocated
4. *GetState*
5. Pickup Food
6. *Get State*
7. Going to Destination
8. *Get State*
9. Complete
10. *Get State*



Writing

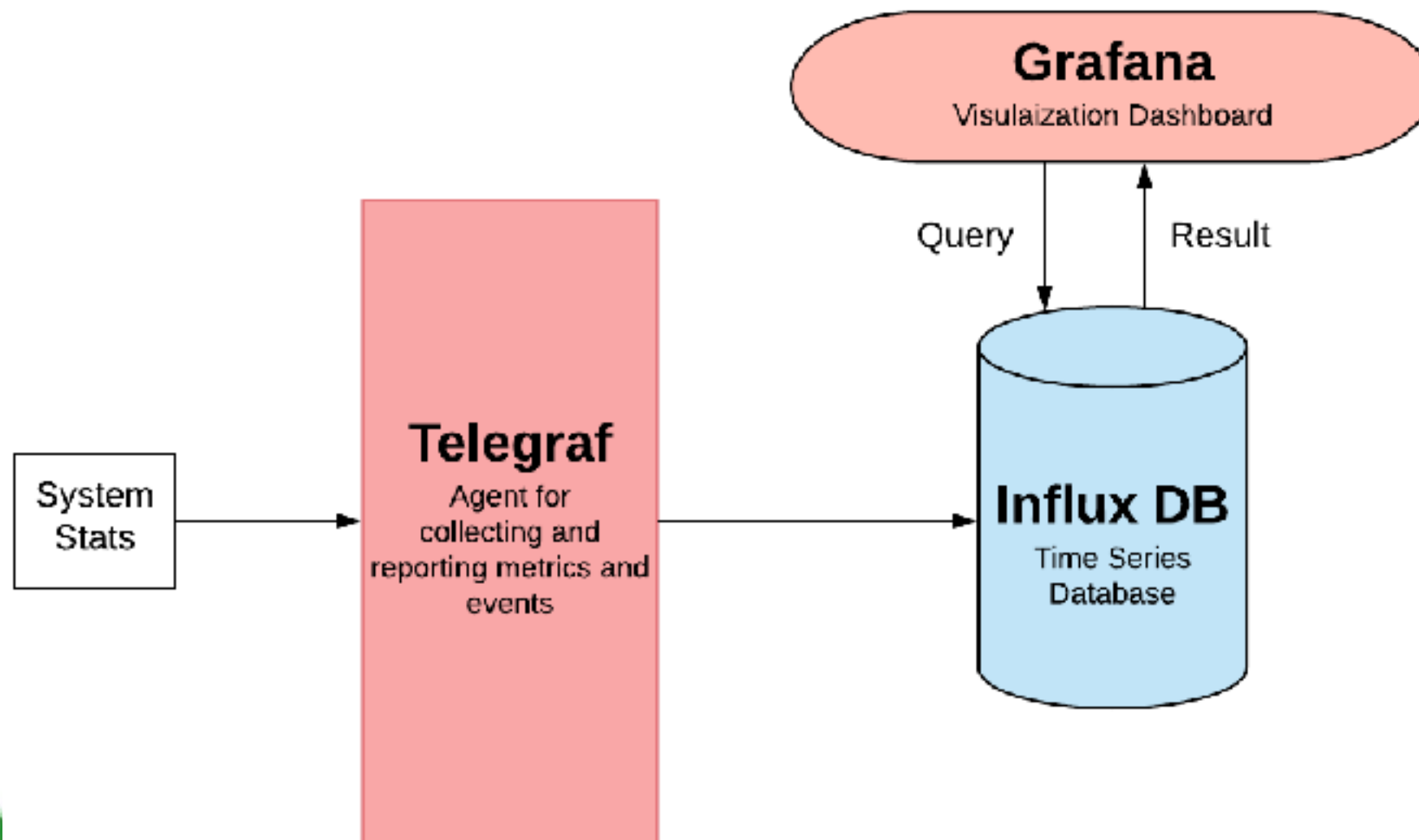


Would you like to know if there is a



Memory leak

Metric Tracking Architecture



Initialise Statsd

```
InitiateStatsDMetrics (config StatsDConfig) {  
    address := fmt.Sprintf("%s:%s", host, port)  
    statsD, err := statsdv2.New(statsdv2.Address(address),  
statsdv2.Prefix(appname))  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```


StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```


StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func StatsDMiddlewareLogger() negroni.HandlerFunc {  
    return negroni.HandlerFunc(func(rw http.ResponseWriter, r *http.Request, next  
http.HandlerFunc) {  
        vars := mux.Vars(r)  
        path := r.URL.Path  
        key := GetKeyStructure(r.URL.Path)  
        t := TimingInStatsD()  
        noOfGoRoutine := runtime.NumGoroutine()  
        next(rw, r)  
        SendInStatsD(key+".time", t)  
        IncrementInStatsD(key + ".calls")  
        GaugeKeyInStatsD(key+".goroutines", noOfGoRoutine)  
    })  
}
```

StatsD Middleware

```
func TimingInStatsD()  
*statsdv2.Timing {  
if statsD == nil {  
return nil  
}  
t := statsD.NewTiming()  
return &t  
}
```

```
func SendInStatsD(key string, t *statsdv2.Timing) bool {  
if statsD == nil {  
return false  
}  
t.Send(key)  
return true  
}
```

```
func IncrementInStatsD(key string) bool {  
if statsD == nil {  
return false  
}  
statsD.Increment(key)  
return true  
}
```

```
func GetKeyStructure(url string) string {  
baseKey := "go.response"  
basePath := strings.Split(url, "/GF")[0]  
keyBasePath := strings.Replace(basePath, "/",  
".", len(basePath))  
key := baseKey + keyBasePath  
return key  
}
```


StatsD Middleware

```
func TimingInStatsD()  
*statsdv2.Timing {  
if statsD == nil {  
return nil  
}  
t := statsD.NewTiming()  
return &t  
}
```

```
func SendInStatsD(key string, t *statsdv2.Timing) bool {  
if statsD == nil {  
return false  
}  
t.Send(key)  
return true  
}
```

```
func IncrementInStatsD(key string) bool {  
if statsD == nil {  
return false  
}  
statsD.Increment(key)  
return true  
}
```

```
func GetKeyStructure(url string) string {  
baseKey := "go.response"  
basePath := strings.Split(url, "/GF")[0]  
keyBasePath := strings.Replace(basePath, "/",  
".", len(basePath))  
key := baseKey + keyBasePath  
return key  
}
```

StatsD Middleware

```
func TimingInStatsD()  
*statsdv2.Timing {  
if statsD == nil {  
return nil  
}  
t := statsD.NewTiming()  
return &t  
}
```

```
func SendInStatsD(key string, t *statsdv2.Timing) bool {  
if statsD == nil {  
return false  
}  
t.Send(key)  
return true  
}
```

```
func IncrementInStatsD(key string) bool {  
if statsD == nil {  
return false  
}  
statsD.Increment(key)  
return true  
}
```

```
func GetKeyStructure(url string) string {  
baseKey := "go.response"  
basePath := strings.Split(url, "/GF")[0]  
keyBasePath := strings.Replace(basePath, "/",  
".", len(basePath))  
key := baseKey + keyBasePath  
return key  
}
```

StatsD Middleware

```
func TimingInStatsD()  
*statsdv2.Timing {  
    if statsD == nil {  
        return nil  
    }  
    t := statsD.NewTiming()  
    return &t  
}
```

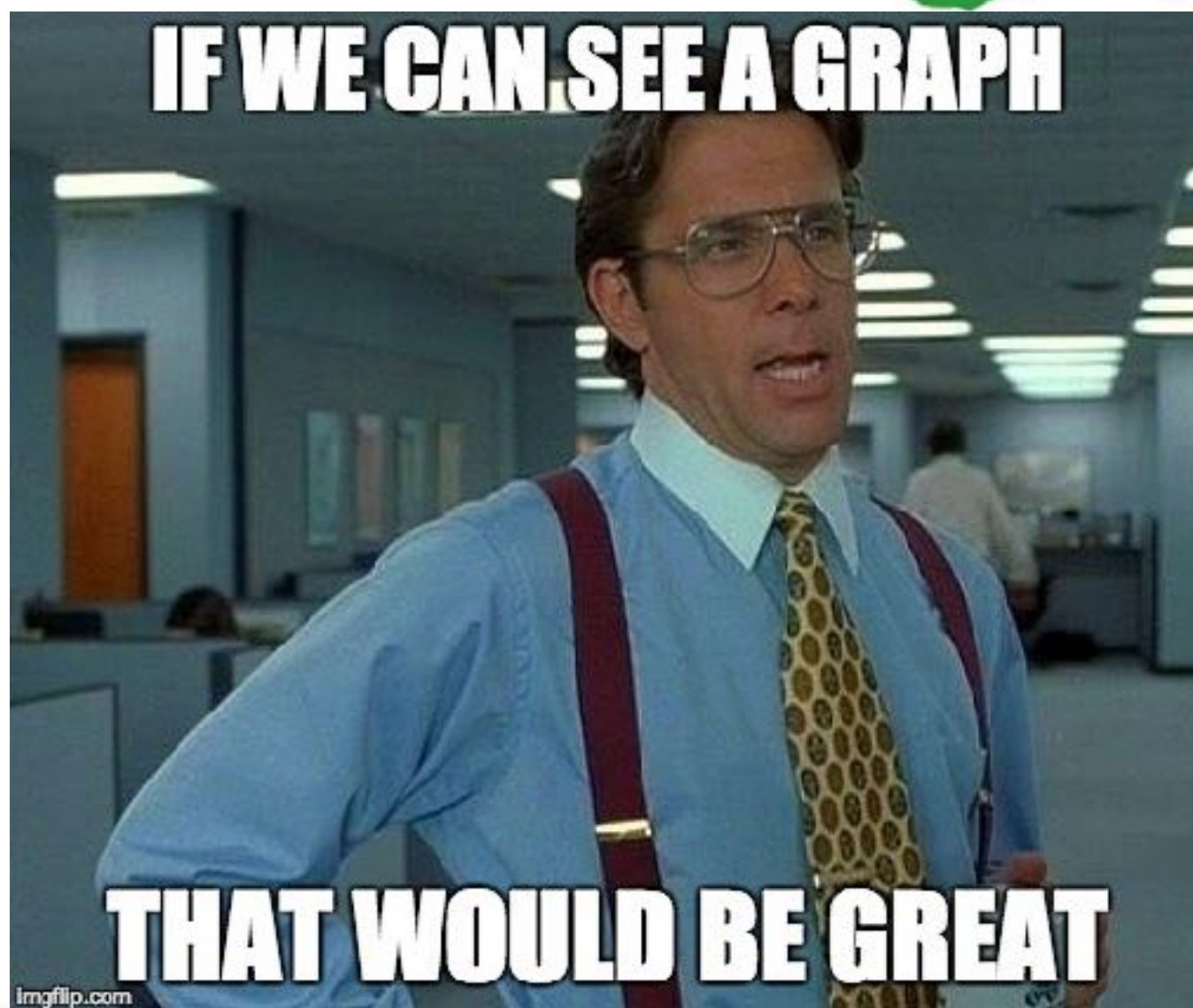
```
func SendInStatsD(key string, t *statsdv2.Timing) bool {  
    if statsD == nil {  
        return false  
    }  
    t.Send(key)  
    return true  
}
```

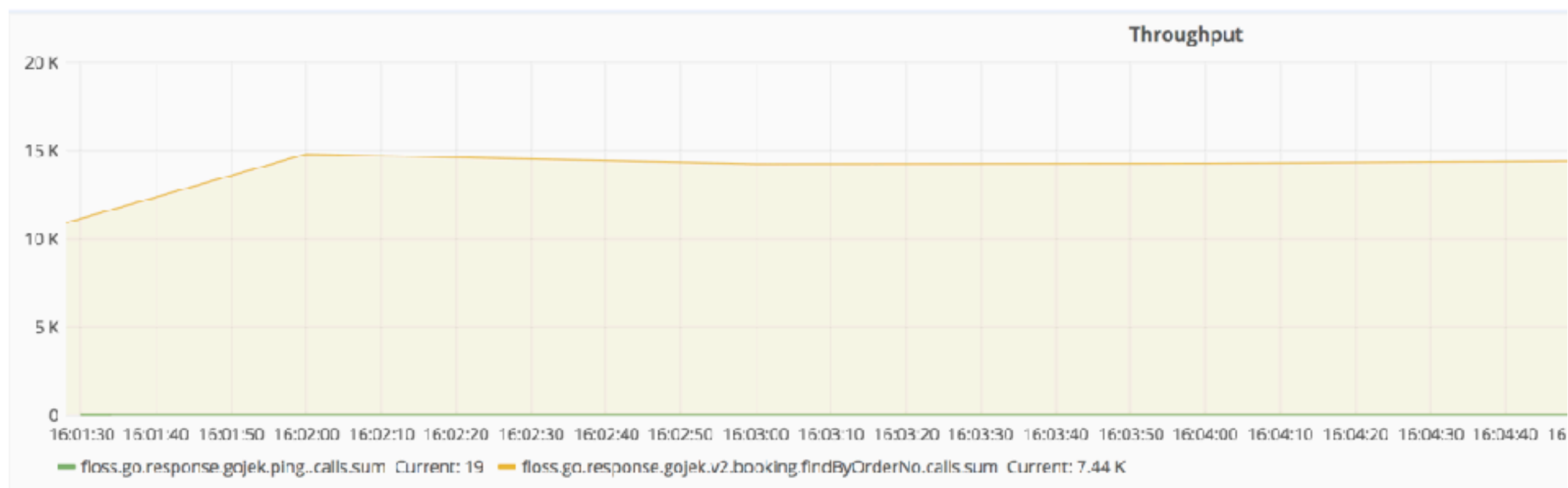
```
func IncrementInStatsD(key string) bool {  
    if statsD == nil {  
        return false  
    }  
    statsD.Increment(key)  
    return true  
}
```

```
func GetKeyStructure(url string) string {  
    baseKey := "go.response"  
    basePath := strings.Split(url, "/GF")[0]  
    keyBasePath := strings.Replace(basePath, "/",  
    ".", len(basePath))  
    key := baseKey + keyBasePath  
    return key  
}
```

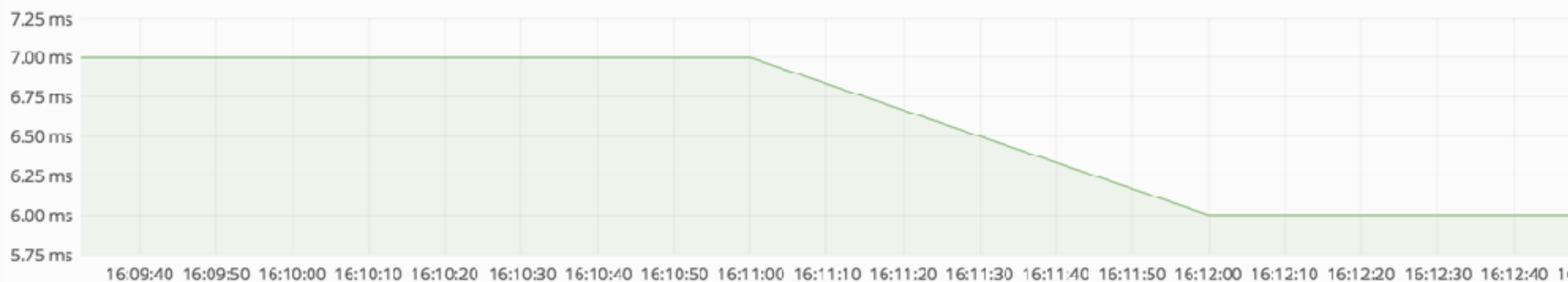
StatsD Middleware

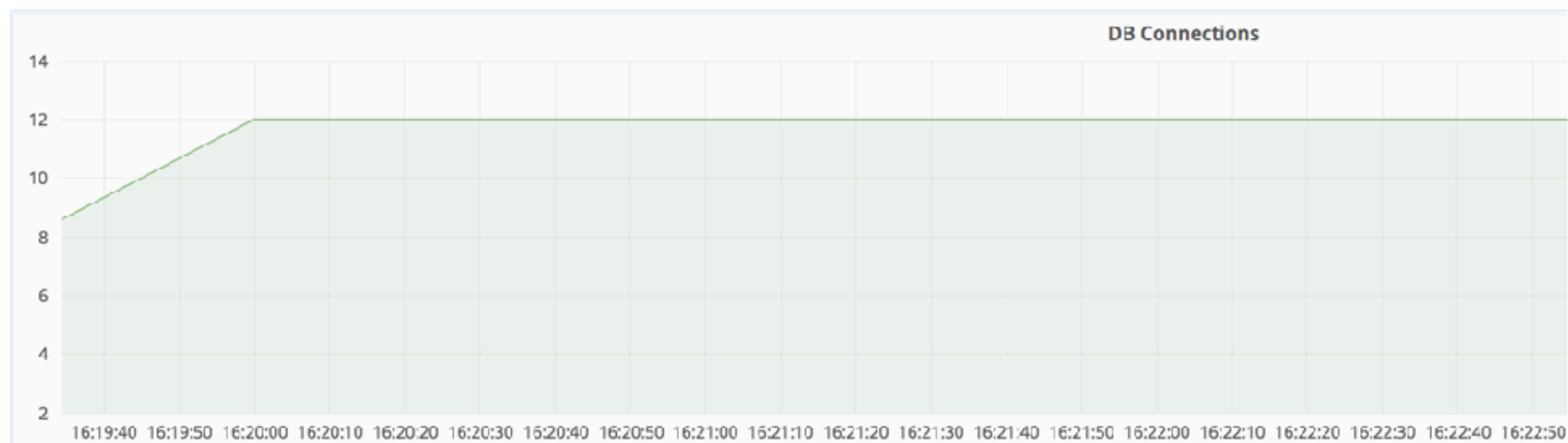
```
n := negroni.New(negroni.NewRecovery())  
n.Use(instrumentation.StatsDMiddlewareLogger())
```

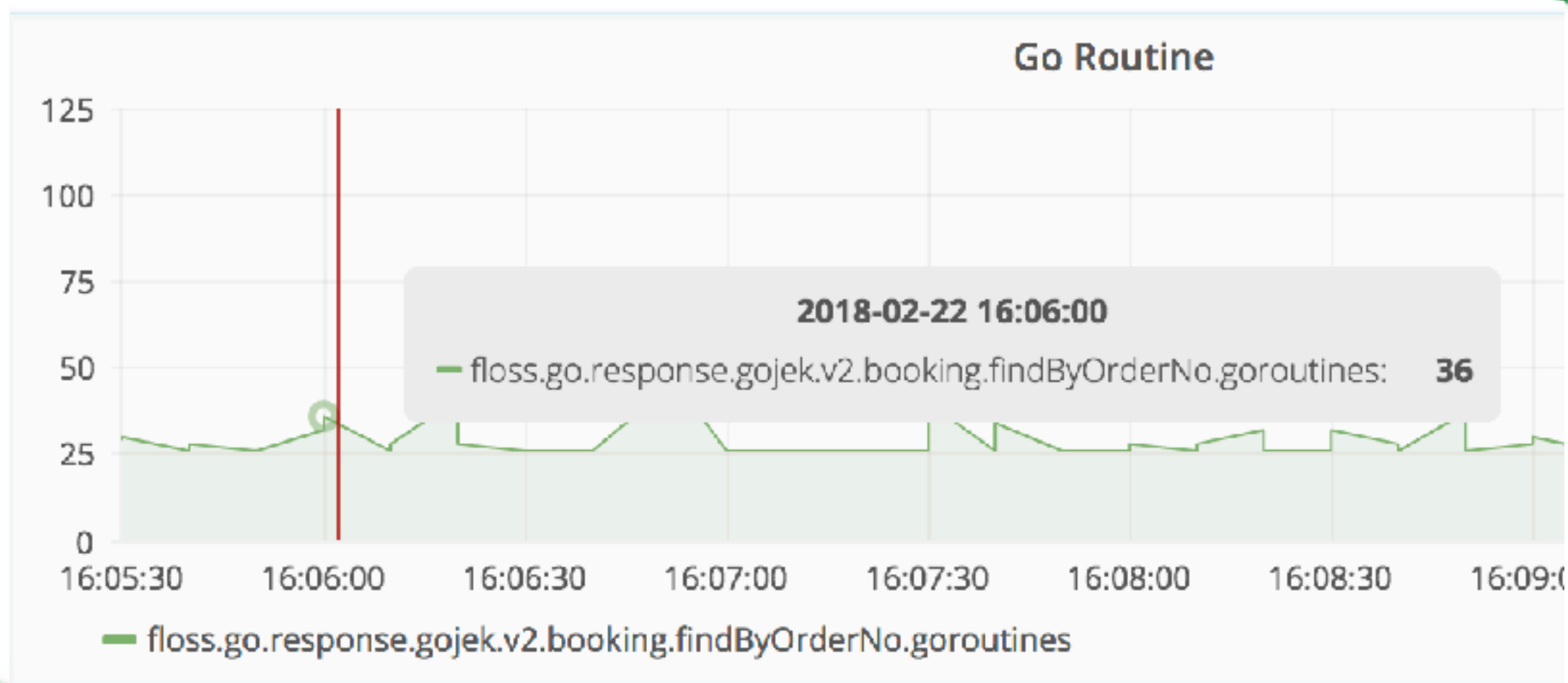





External Services







New Relic Instrumentations

```
func StartDataSegmentNowForDataStore(op string, tableName string,
txn newrelic.Transaction, datastore newrelic.DatastoreProduct)
newrelic.DatastoreSegment {
    s := newrelic.DatastoreSegment{
        Product:    datastore,
        Collection:  tableName,
        Operation:   op,
    }

    s.StartTime = newrelic.StartSegmentNow(txn)
    return s
}
```


New Relic Instrumentations

```
func StartDataSegmentNowForDataStore(op string, tableName string,  
txn newrelic.Transaction, datastore newrelic.DatastoreProduct)  
newrelic.DatastoreSegment {  
    s := newrelic.DatastoreSegment{  
        Product:    datastore,  
        Collection: tableName,  
        Operation:  op,  
    }  
  
    s.StartTime = newrelic.StartSegmentNow(txn)  
    return s  
}
```

New Relic Instrumentations

```
func StartDataSegmentNowForDataStore(op string, tableName string,
txn newrelic.Transaction, datastore newrelic.DatastoreProduct)
newrelic.DatastoreSegment {
    s := newrelic.DatastoreSegment{
        Product:    datastore,
        Collection: tableName,
        Operation:  op,
    }

    s.StartTime = newrelic.StartSegmentNow(txn)
    return s
}
```

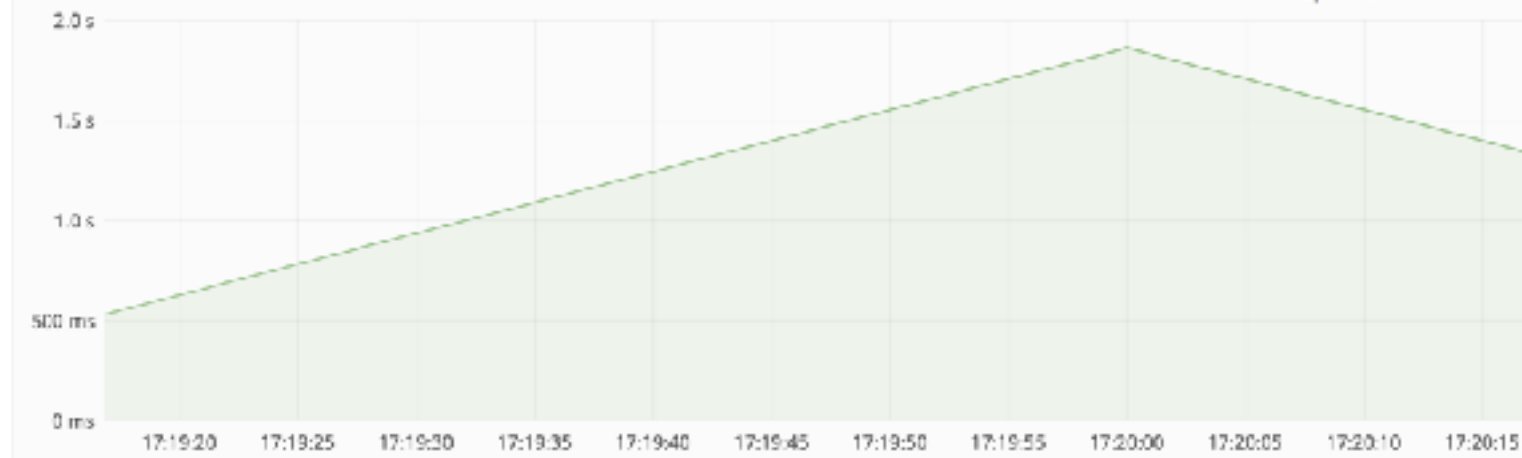
| Name | Vendor | Operation | Collection | Call count | Throughput (cpm) | Avg (ms) | Min (ms) | Max (ms) |
|--|--------|------------------------------------|------------|------------|------------------|----------|----------|----------|
| MySQL findBookingQuery | MySQL | findBookingQuery | (all) | 125,000 | 4,150 | 0.025 | 0.001 | 1.89 |
| MySQL Booking findBookingQuery | MySQL | findBookingQuery | (all) | 125,000 | 4,150 | 0.025 | 0.001 | 1.89 |
| MySQL findRouteItemRepository | MySQL | findRouteItemRepository | (all) | 114,000 | 3,800 | 0.008 | 0.001 | 1.73 |
| MySQL route_item findRouteItemRepository | MySQL | findRouteItemRepository | (all) | 114,000 | 3,800 | 0.008 | 0.001 | 1.73 |
| MySQL findAddressQuery | MySQL | findAddressQuery | (all) | 113,000 | 3,760 | 0.006 | 0.001 | 1.69 |
| MySQL booking_route findAddressQuery | MySQL | findAddressQuery | (all) | 113,000 | 3,760 | 0.006 | 0.001 | 1.69 |
| MySQL booking_driver_cancellation_log findBookingDriverCancellationQuery | MySQL | findBookingDriverCancellationQuery | (all) | 114,000 | 3,790 | 0.006 | 0.001 | 1.97 |
| MySQL findBookingDriverCancellationQuery | MySQL | findBookingDriverCancellationQuery | (all) | 114,000 | 3,790 | 0.006 | 0.001 | 1.97 |
| MySQL pricing_info findPriceInfoQuery | MySQL | findPriceInfoQuery | (all) | 20,600 | 686 | 0.003 | 0.001 | 0.903 |
| MySQL findPriceInfoQuery | MySQL | findPriceInfoQuery | (all) | 20,600 | 686 | 0.003 | 0.001 | 0.903 |

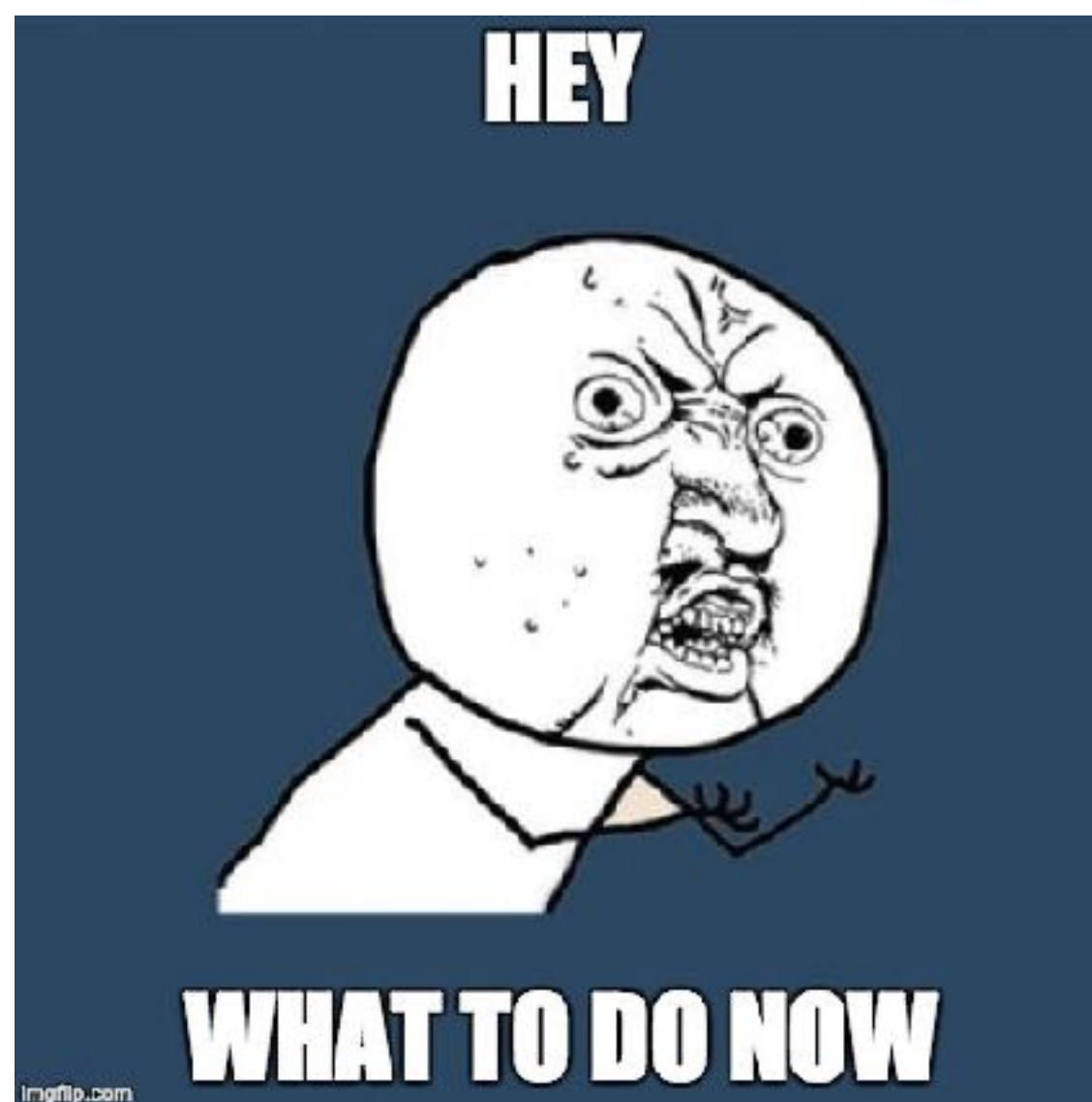


Go Routine



Response Time






```
import "net/http/pprof"
```

```
func Router(dependencies *service.Instances) *mux.Router {  
    router := mux.NewRouter()  
    router.HandleFunc("/debug/pprof/", pprof.Index)  
    router.HandleFunc("/debug/pprof/cmdline", pprof.Cmdline)  
    router.HandleFunc("/debug/pprof/profile", pprof.Profile)  
    router.HandleFunc("/debug/pprof/symbol", pprof.Symbol)  
    router.HandleFunc("/debug/pprof/trace", pprof.Trace)  
    router.HandleFunc("/debug/pprof/goroutine", pprof.Index)  
    return router  
}
```

```
import "net/http/pprof"
```

```
func Router(dependencies *service.Instances) *mux.Router {  
    router := mux.NewRouter()  
    router.HandleFunc("/debug/pprof/", pprof.Index)  
    router.HandleFunc("/debug/pprof/cmdline", pprof.Cmdline)  
    router.HandleFunc("/debug/pprof/profile", pprof.Profile)  
    router.HandleFunc("/debug/pprof/symbol", pprof.Symbol)  
    router.HandleFunc("/debug/pprof/trace", pprof.Trace)  
    router.HandleFunc("/debug/pprof/goroutine", pprof.Index)  
    return router  
}
```

/debug/pprof/

profiles:

0 [block](#)

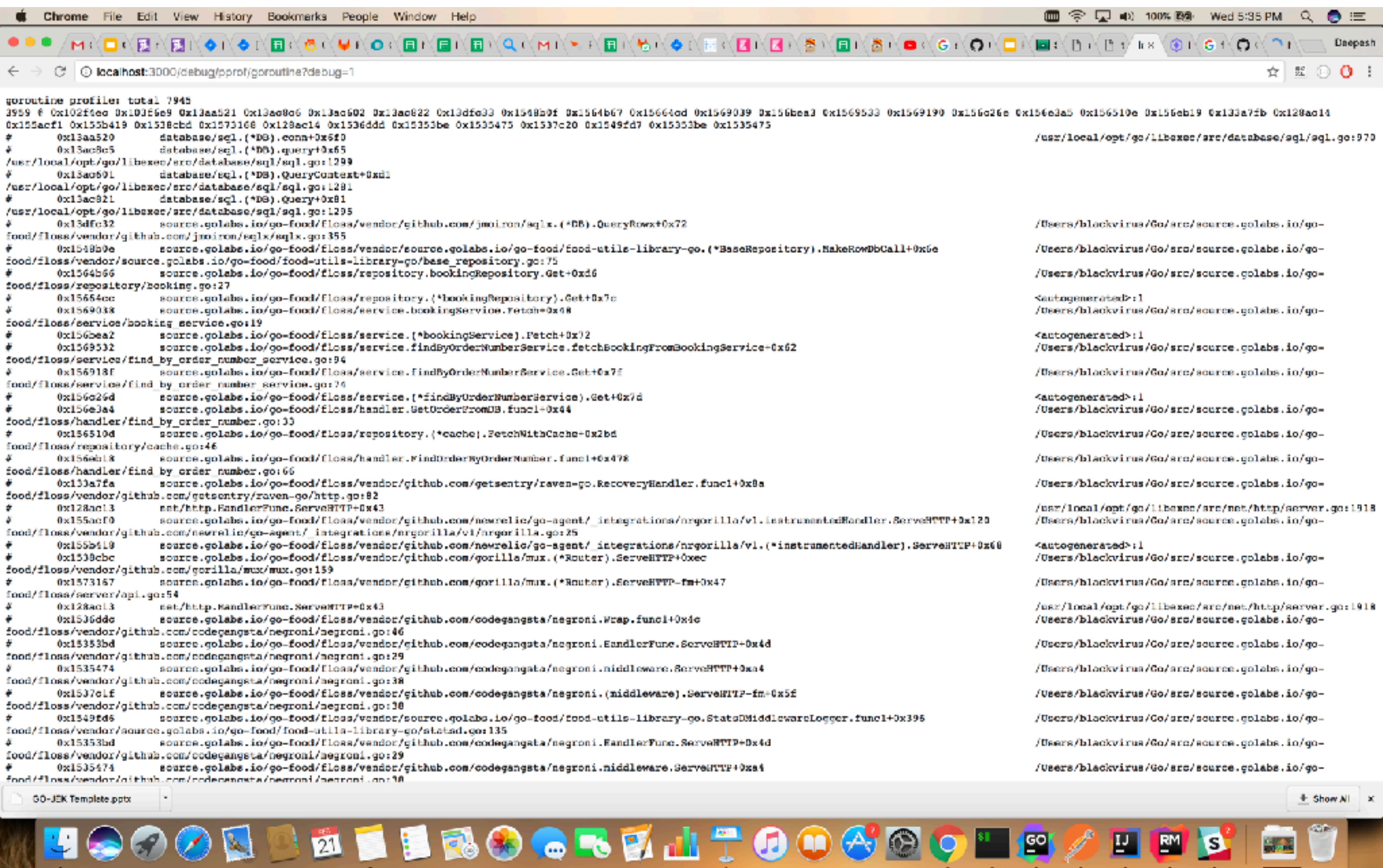
799 [goroutine](#)

72 [heap](#)

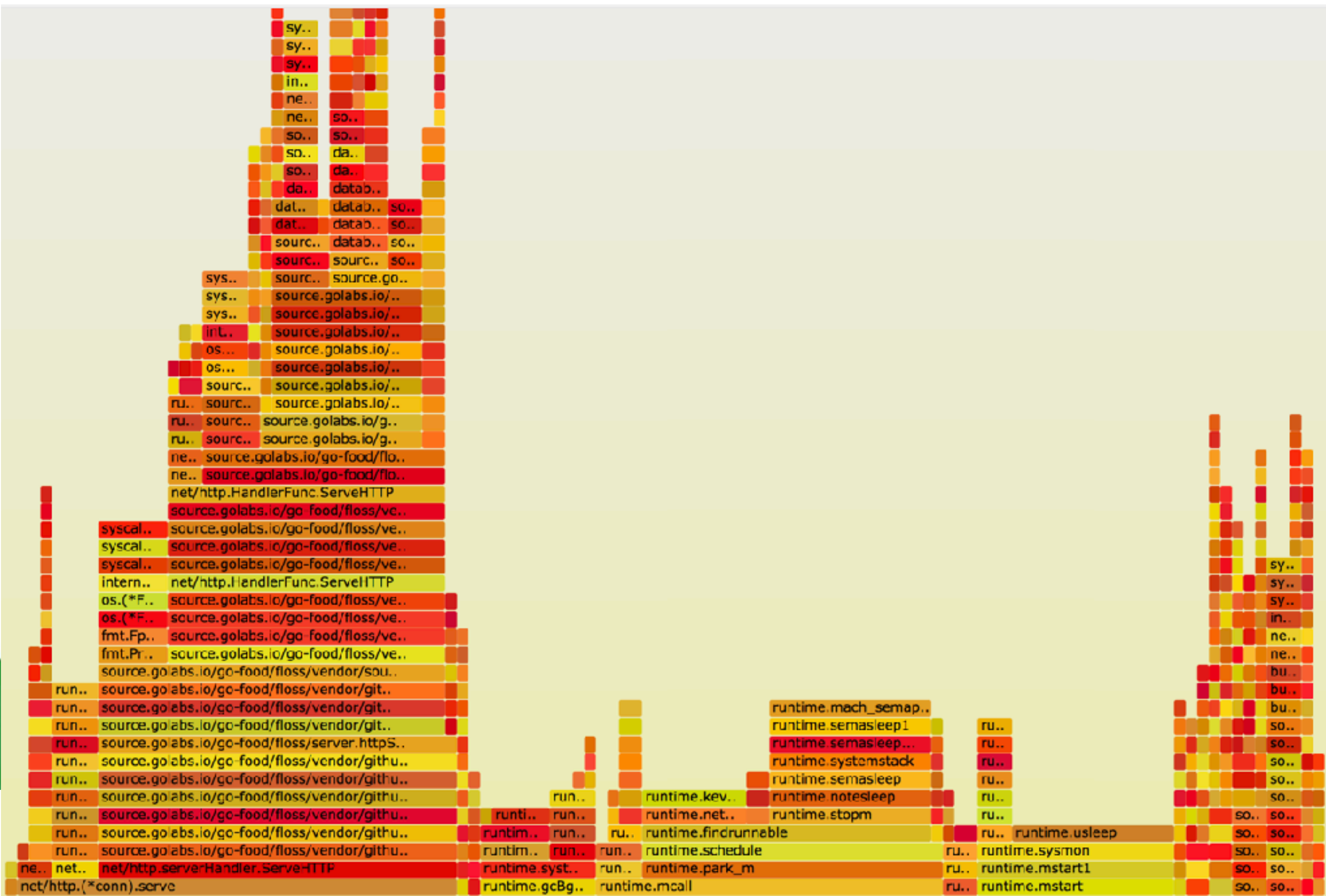
0 [mutex](#)

13 [threadcreate](#)

[full goroutine stack dump](#)



go tool pprof -raw -seconds 10 <http://localhost:3000/debug/pprof/profile>

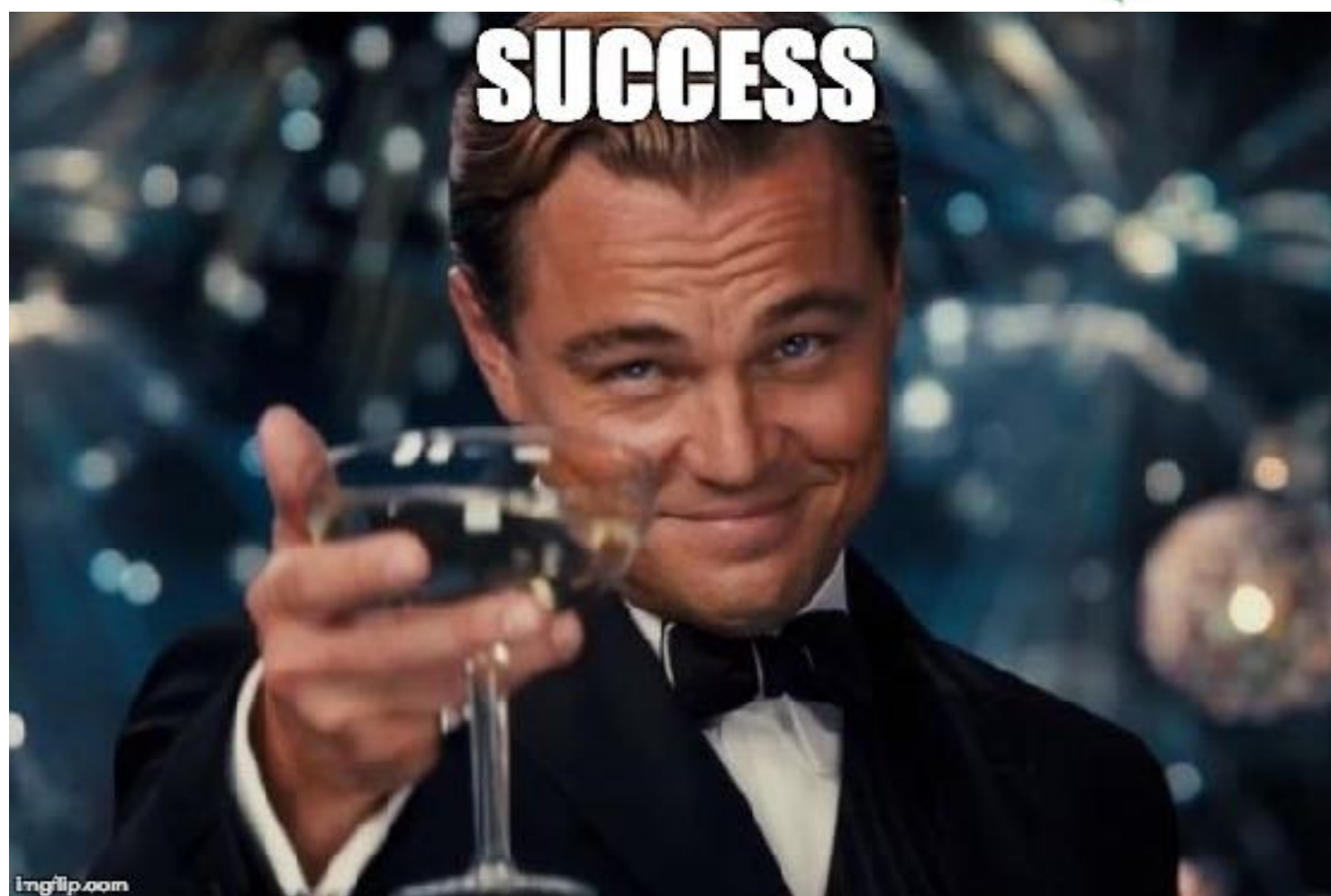


```
go tool pprof -raw -seconds 10 http://localhost:3000/debug/pprof/profile
```

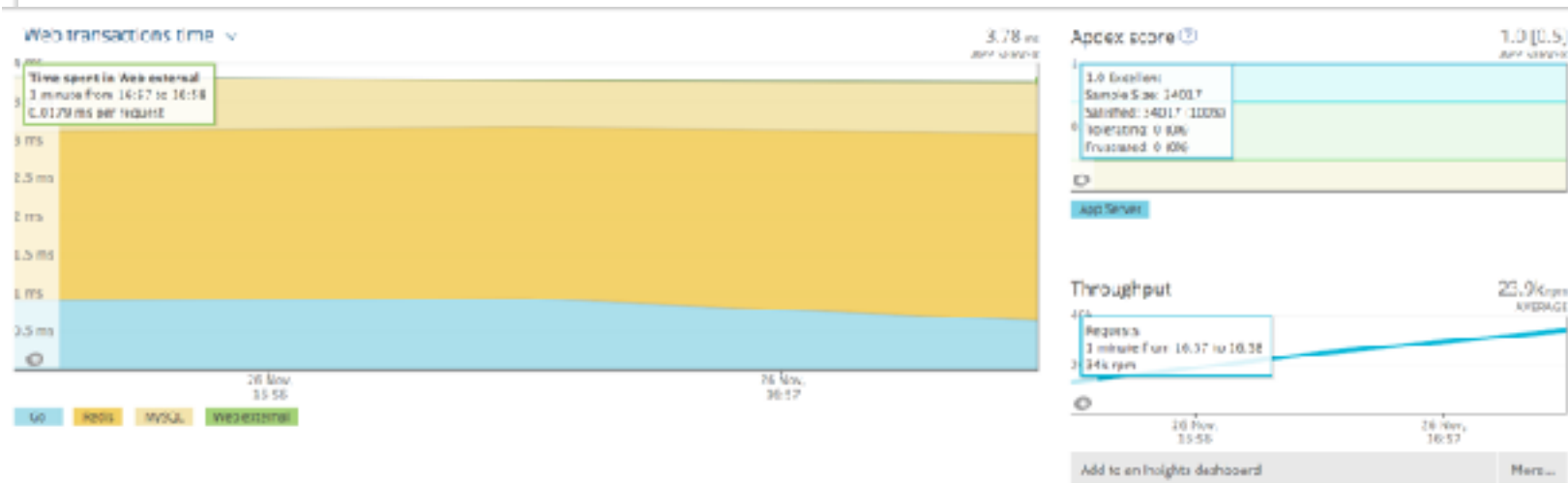
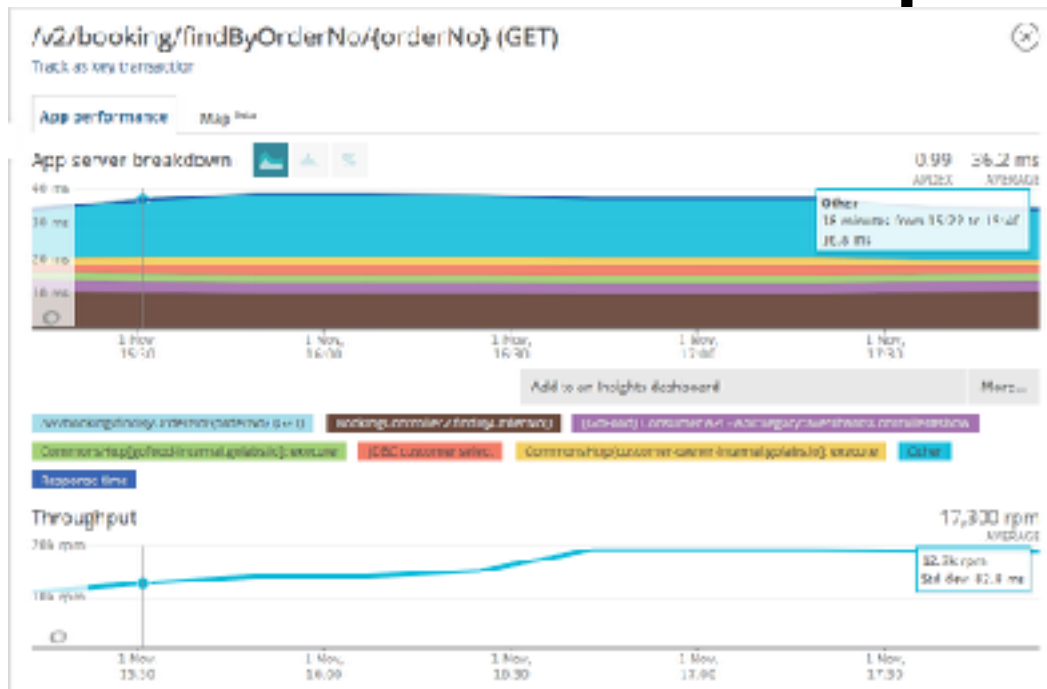



```
return rows.Next()
```

```
boolean := false
defer func() {
if rows != nil {
rows.Close()
}
}()
if rows.Next() {
boolean = true
}
return boolean, nil
```



Reduction in response times



Things to Remember

Metrics

Profiling

Monitoring

Things to Remember

Metrics

Profiling

Monitoring

Things to Remember

Metrics

Profiling

Monitoring

Thank You !

<https://github.com/blackvirus18/gophercon2018>