

CASE DE MIGRAÇÃO

# Microservice de cálculo de Node.js para Golang

**HARMO**

Uma empresa **ReclameAQUI**



## Leonardo Rifeli

Co-founder & CTO



*Sempre vai ter alguém que  
vai aprender muito com o  
pouco que você sabe. Se  
você compartilhar, é claro.*

- 27 years ago
- Data Science - Uninter
- Developer since 2010

- Gopher since 2017 (v1.7)
- CTO at harmo.me
- Mentor at ACATE

**Agora a  
experiência é o  
novo marketing**





# HARMO

Geramos **tráfego orgânico** para lojas físicas e online, através do incentivo ao boca a boca online.



**SEO Local + Reviews + Pesquisas**

Tudo integrado em uma única solução focada em aquisição orgânica de clientes.

+70 grandes marcas estão diminuindo o CAC com a Harmo

 Ipiranga

 C&A

 americanas

 HERING

 makro

 Bob's

 ATACADÃO

 ampm

 DROGARIA  
catarinense  
A gente cuida de você

 FARMÁCIA  
PREÇO POPULAR

 CINEMARK

 super nosso  
GOURMET DE CORAÇÃO

 Savegnago

 lopes  
supermercados

 ALMEIDA JUNIOR

 FARMARCAS

 Mania de churrasco!  
PRIME STEAK HOUSE

 Pato Corrente  
PRIME STEAK HOUSE

 MR.CAT

 SAPHYR  
SHOPPING CENTERS

 CAFÉ  
CULTURA

 DETROIT  
STEAKHOUSE

 CASSOL  
CENTERLAR

 ITAIPU  
BINACIONAL

 FLORIPA  
SHOPPING

# Números

Establishments

**+40k**

Integrations

**+60k**

Reviews

**+15mi**

Emails

**+16mi**

SMS

**+1.7mi**

Review Answers

**+2mi**

# Topics

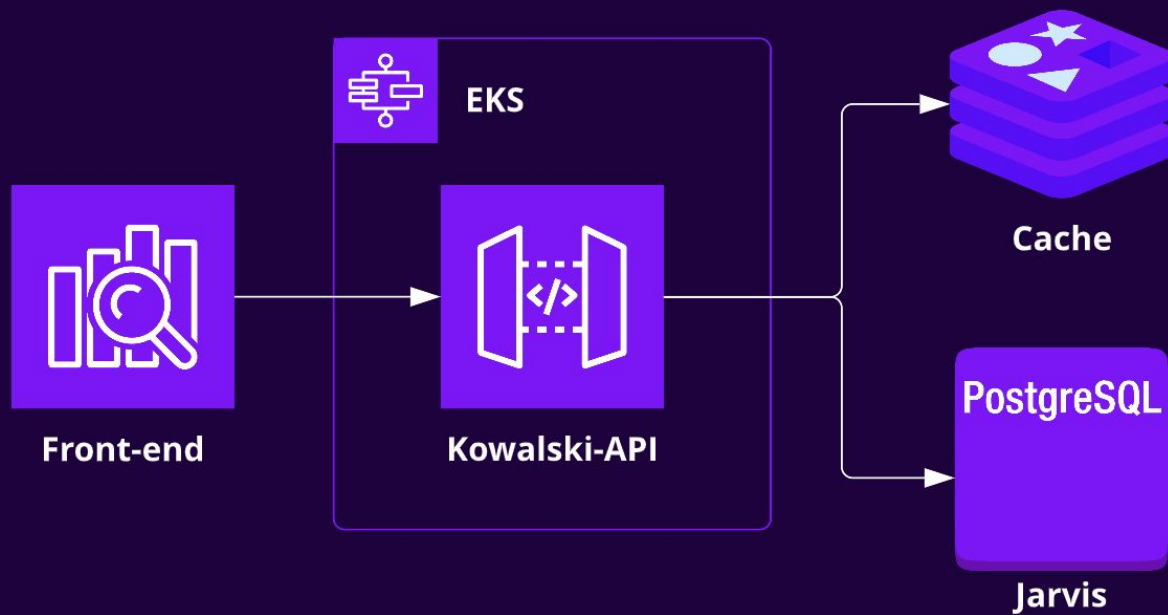
- RRI Equation
- Explain Endpoint & Parsers
- About Node.js (threads?)
- About Golang
- Implementations
- Results
- Conclusion

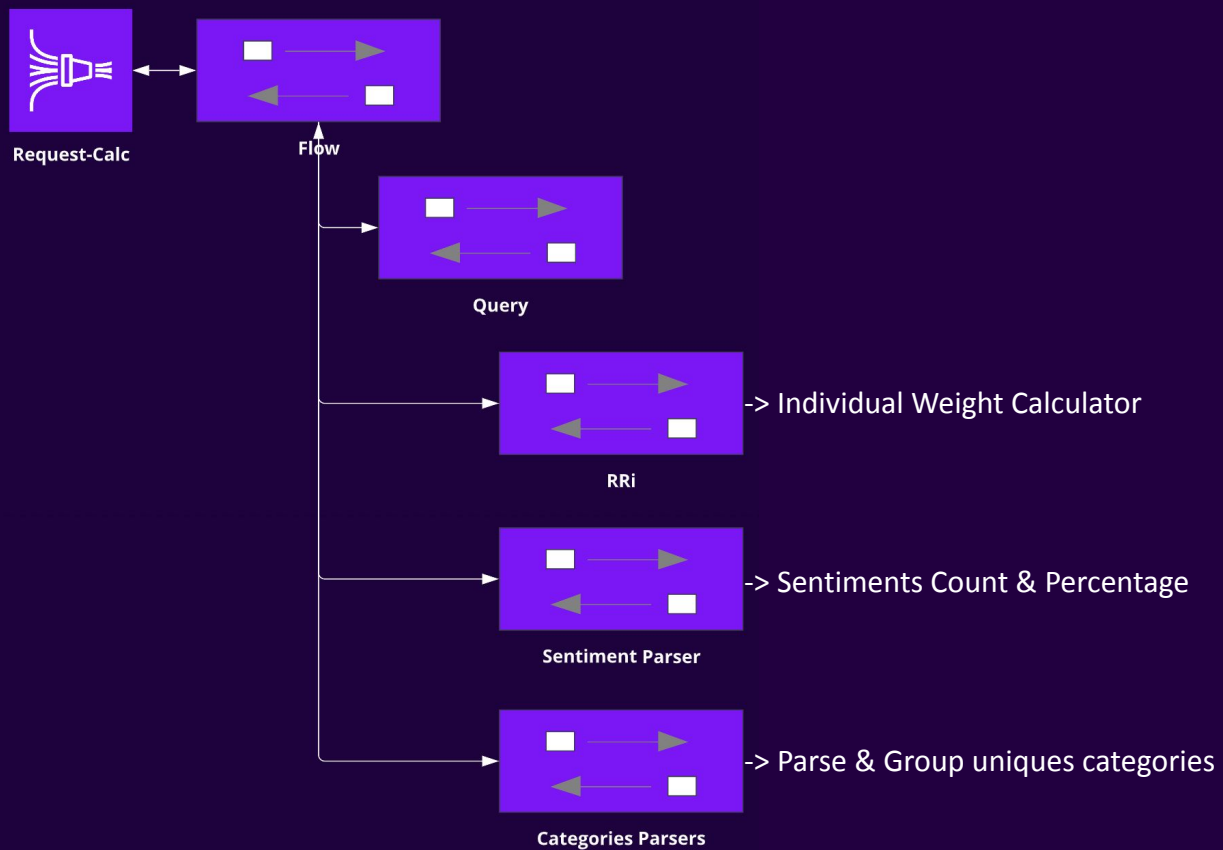
# $RR_i$ Equation



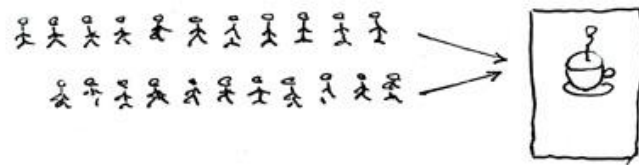
$$ir_{jkt} = \frac{\sum_{i \in I_{jt}} \left( \frac{s_{ijk}}{\ln(2 + \Delta t_{ijk})} \right)}{\sum_{i \in I_{jt}} \left( \frac{1}{\ln(2 + \Delta t_{ijk})} \right)}$$

# Explain Endpoint & Parsers

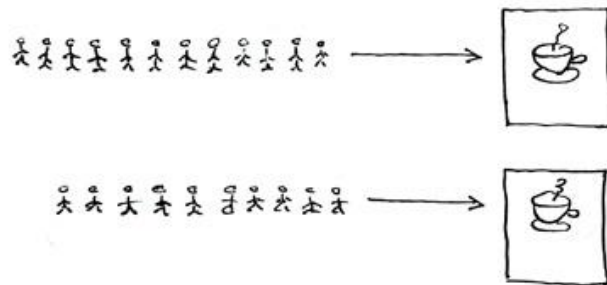




Concurrent = Two Queues One Coffee Machine

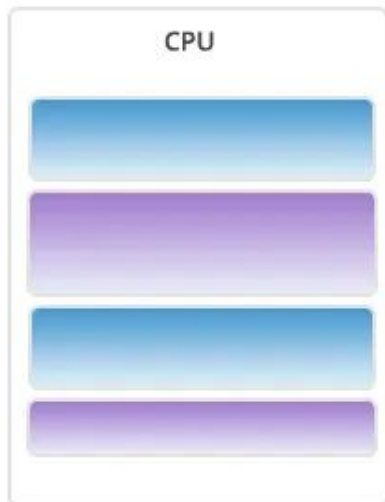


Parallel = Two Queues Two Coffee Machines





## Concurrency



## Parallelism

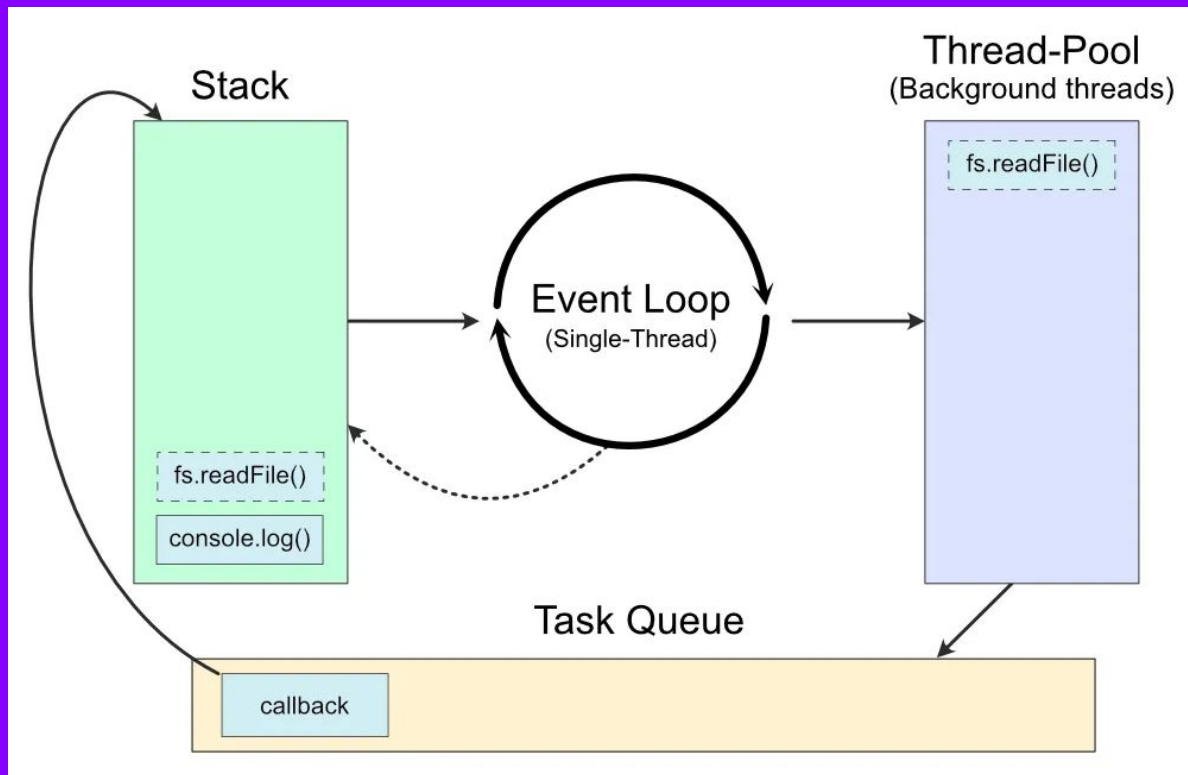


# Node.js

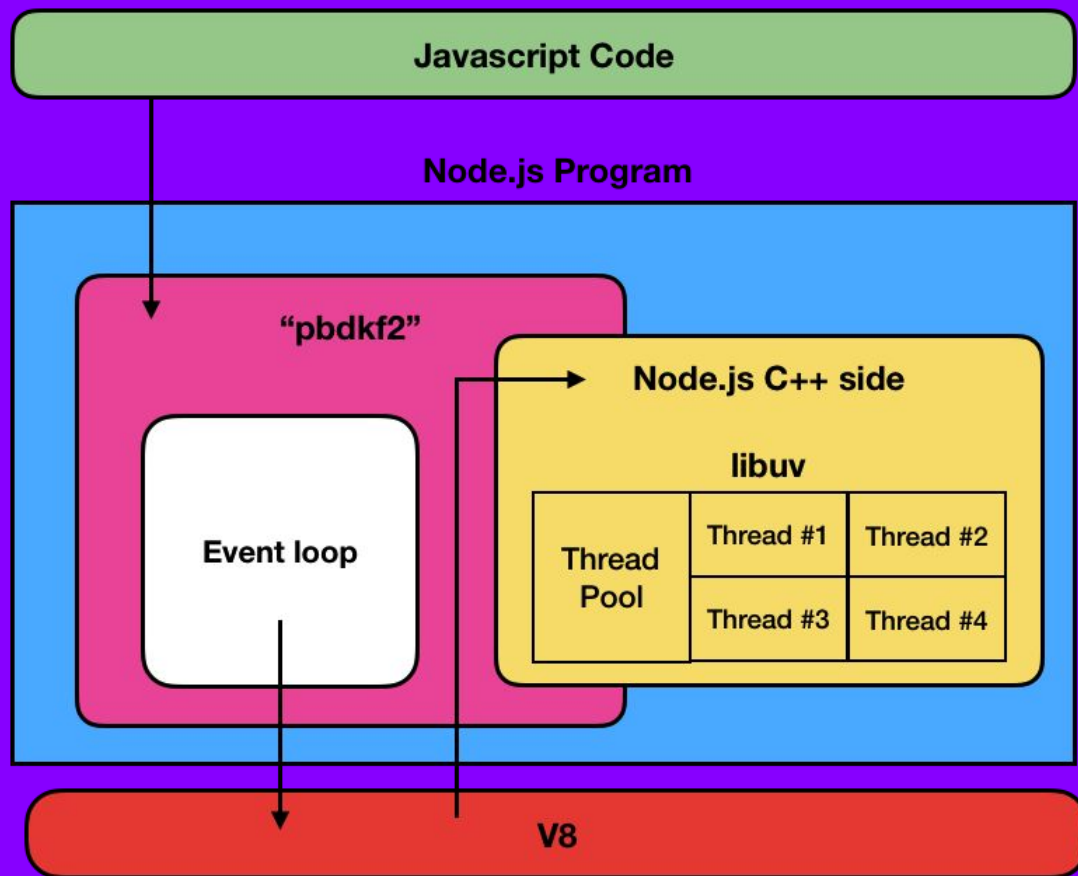
***Node.js is a JavaScript***  
**runtime built on Chrome's V8**  
**JavaScript engine**

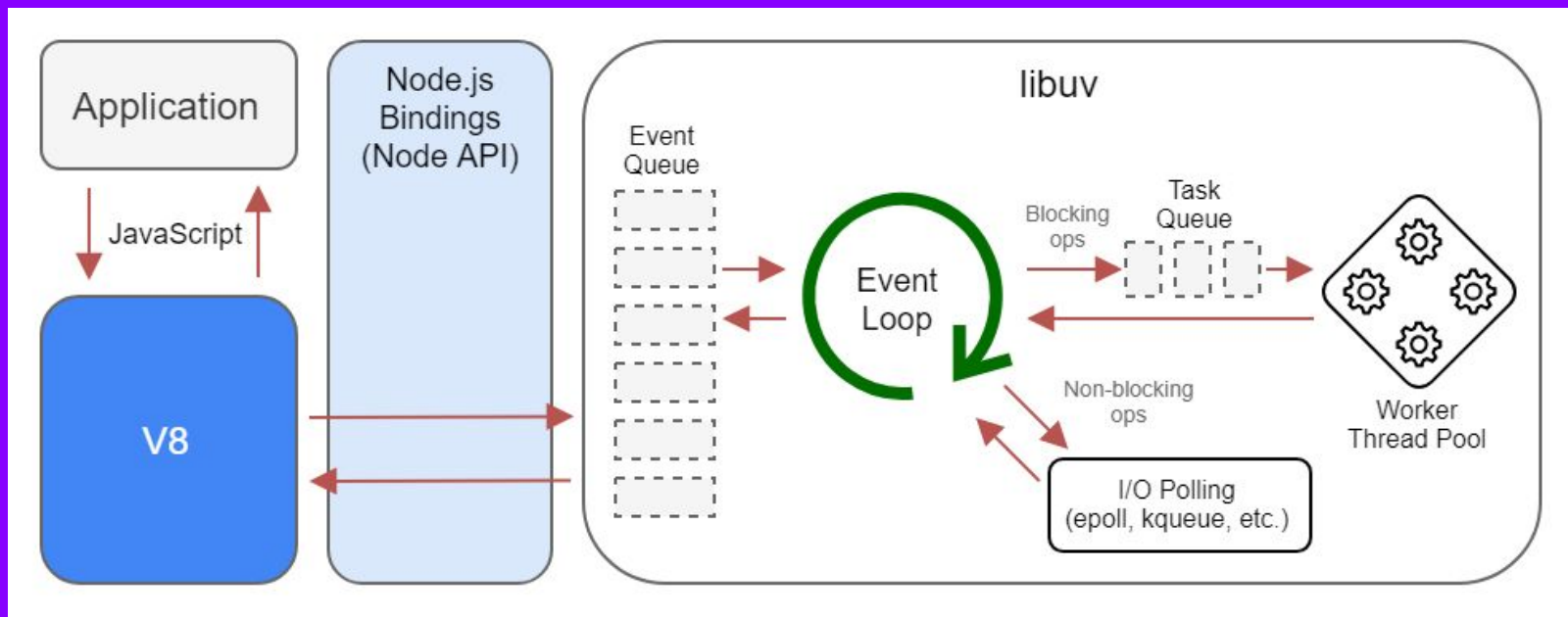


***Ryan Dahl***  
***developed it in 2009***









**Kowalski-API was born**  
***with Node.js 7.9.0*** (tudo era mato)

**node:worker\_thread**  
***became available v10.5***

# Golang



# Why Golang?

- **We use Golang since 2017 (v1.7)**
- **Efficient in compilation and execution**
- **Massive data processing over the network (example Google)**
- **Excellent memory management**
- **Inheritance never more \o/**
- **Generic purpose: Machine learning, IoT, databases, microservices**

**First Release at 2009-11-10**

**24 hours later**  
***First Comment About***

Robert Griesemer  
(dev Google)

Rob Pike (Unix)

Ken Thompson  
(Unix, B, C)





# Simple *Semantic*

**Is a statically-typed**  
***Language***

# Concurrent & Parallel *Language*

# Packages

```
package main

import (
    "fmt"
    "math"
)

func main() {
    fmt.Printf("Now you have %g problems.", math.Sqrt(7))
}
```

# **Multiples** ***Results***

```
package main

import "fmt"

func swap(x, y string) (string, string) {
    return y, x
}

func main() {
    a, b := swap("hello", "world")
    fmt.Println(a, b)
}
```

# Errors



```
package main

import "github.com/coderockr/nfe/transmitter"

func main() {
    response, err := transmitter.transmit(nfe, xml)
    if err != nil {
        panic("Error ") //tratamento de erro qualquer
    }
    result, err := transmitter.saveData(response, xml)
    if err != nil {
        panic("Error ") //tratamento de erro qualquer
    }
}
```

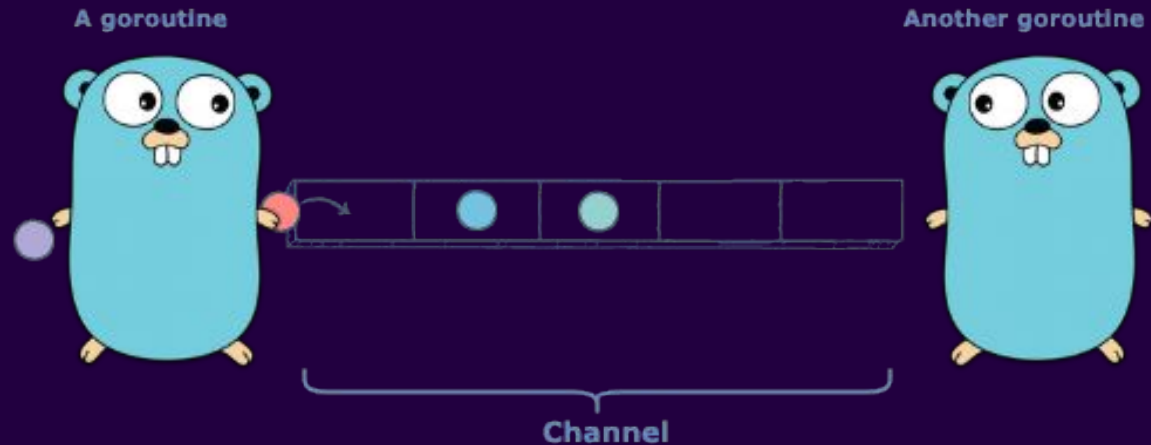
# Goroutines \o/

# The Go Playground

```
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 func say(s string) {
9     for i := 0; i < 5; i++ {
10         time.Sleep(100 * time.Millisecond)
11         fmt.Println(s)
12     }
13 }
14
15 func main() {
16     go say("Acate routine")
17     say("Single")
18     say("Single-Other")
19 }
```

```
Single
Acate routine
Acate routine
Single
Single
Acate routine
Acate routine
Single
Single
Acate routine
Single-Other
Single-Other
Single-Other
Single-Other
Single-Other
```

**Channels \o/**



**CSP Protocol (Communicating Sequential Processes)**  
**Or**  
**Goroutines**  
**\*\*No shared memory state\*\***

# The Go Playground

```
1 package main
2
3 import (
4     "fmt"
5     "time"
6 )
7
8 func say(s string, done chan string) {
9     for i := 0; i < 5; i++ {
10         time.Sleep(100 * time.Millisecond)
11         fmt.Println(s)
12     }
13
14     done <- "Done!!!"
15 }
16
17
18 func main() {
19     done := make(chan string)
20     go say("Acate", done)
21     fmt.Println(<-done)
22 }
```



Acate  
Acate  
Acate  
Acate  
Acate  
Done!!!

Program exited.

# Blocking *Syntax*

```
1  package main
2
3  import "fmt"
4
5  v func main() {
6      initChan := make(chan int)
7
8      initChan <- 10 //it blocks
9
10     rec := <-initChan //it blocks
11
12     fmt.Printf("Received Value: %d\n", rec)
13 }
```



~/projects/harmo/talk on  refactored-flow!  18:20:57

⊗ \$ go run block.go

fatal error: all goroutines are asleep - deadlock!









goroutine 1 [chan send]:


main.main()

        /Users/leonardorifeli/projects/harmo/talk/block.go:8 +0x38

exit status 2

# Implementations

Tx: Auto Playground

1		<code>SELECT count(*) FROM review r;</code>
2		<code>-- count: 12 815 545</code>

# Results

# Node.js

## *Benchmark*

```
~/projects/harmo/talk/wolframalpha-nodejs-async on [?] main 15:48:07
$ node index.mjs
-> Start!
Query: 6.786s
Calc_RRi: 1.707s
Parse_Categories: 10.980s
Parse_Sentiments: 107.413ms
-> Final RRi: 8.7507
-> Final Sentiments: {
  positive: 3315891,
  positive_percent: 84.16,
  neutral: 319444,
  neutral_percent: 8.11,
  negative: 304789,
  negative_percent: 7.74,
  total: 3940124
}
-> Final Categories (66): [
  'cinema', 'lugar', 'praca de alimentacao',
  'servico', 'telefone', 'atendimento',
  'caixa', 'experiencia', 'modelagem',
  'organizacao', 'preco', 'qualidade',
  'servicos financeiros', 'delivery', 'limpeza',
  'comida', 'lojas', 'recepcao',
  'alimentacao', 'produtos', 'lazer',
  'custo beneficio', 'estrutura', 'estacionamento',
  'comodidades', 'combustivel', 'cardapio',
  'bebida', 'amenities', 'eventos',
  'filmes', 'financeiro', 'ingressos',
  'localizacao', 'wi-fi', 'carro',
  'servico', 'governanca', 'provador',
  'aplicativo', 'conveniencia', 'manutencao',
  'higiene', 'graduacao', 'entrega',
  'aulas', 'funcionamento', 'chocolate',
  'opcoes', 'moda', 'pos graduacao',
  'alimentos', 'entreterimento', 'acessorios',
  'agro', 'beleza e saude', 'eletrodomesticos',
  'eletroportateis', 'moveis', 'telefones e celulares',
  'TV e video', 'audio', 'shows',
  'automotivo', 'pecas', 'bebidas'
]
Start: 19.583s
```

```
~/projects/harmo/talk/wolframalpha-nodejs-workers on [?] main 15:54:43
$ node index.mjs
-> Start!
Query: 6.644s
Running Parse_Categories
Running Parse_Sentiments
Parse_Sentiments: 105.796ms
Running Calc_RRi
Calc_RRi: 1.687s
-> Final RRi: 8.7507
-> Final Sentiments: {
  positive: 3315891,
  positive_percent: 84.16,
  neutral: 319444,
  neutral_percent: 8.11,
  negative: 304789,
  negative_percent: 7.74,
  total: 3940124
}
-> Final Categories (66): [
  'cinema', 'lugar', 'praca de alimentacao',
  'servico', 'telefone', 'atendimento',
  'caixa', 'experiencia', 'modelagem',
  'organizacao', 'preco', 'qualidade',
  'servicos financeiros', 'delivery', 'limpeza',
  'comida', 'lojas', 'recepcao',
  'alimentacao', 'produtos', 'lazer',
  'custo beneficio', 'estrutura', 'estacionamento',
  'comodidades', 'combustivel', 'cardapio',
  'bebida', 'amenities', 'eventos',
  'filmes', 'financeiro', 'ingressos',
  'localizacao', 'wi-fi', 'carro',
  'servico', 'governanca', 'provador',
  'aplicativo', 'conveniencia', 'manutencao',
  'higiene', 'graduacao', 'entrega',
  'aulas', 'funcionamento', 'chocolate',
  'opcoes', 'moda', 'pos graduacao',
  'alimentos', 'entreterimento', 'acessorios',
  'agro', 'beleza e saude', 'eletrodomesticos',
  'eletroportateis', 'moveis', 'telefones e celulares',
  'TV e video', 'audio', 'shows',
  'automotivo', 'pecas', 'bebidas'
]
Start: 23.776s
Parse_Categories: 11.014s
```

# Golang

## *Benchmark*

```
~/projects/harmo/talk/wolframalpha-golang on [?]main! 15:57:30
$ time go run main.go
-> Start!

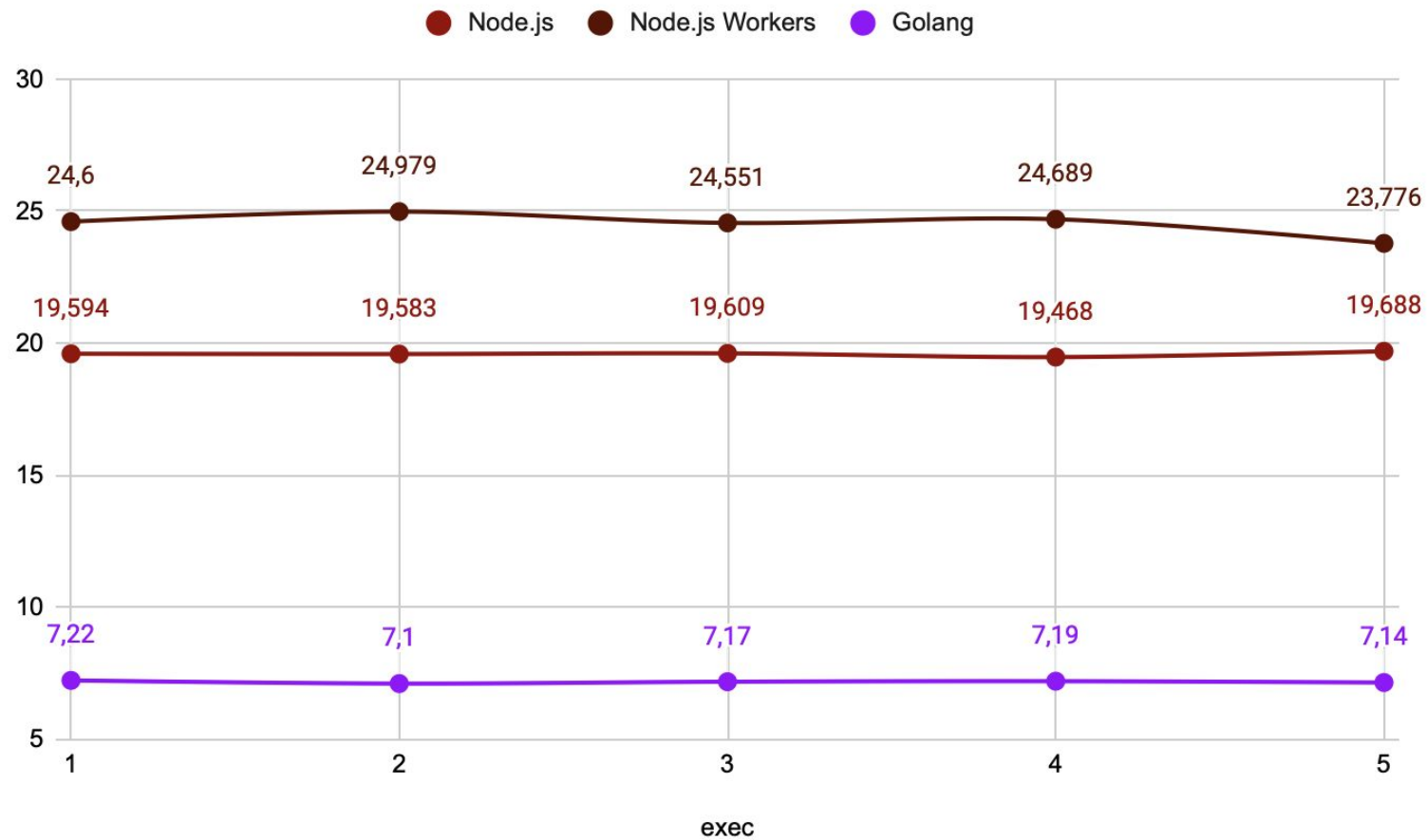
-> Final Sentiments:
Positive: 3315891
Percent: 84%
-----
Neutral: 319444
Percent: 8%
-----
Negative: 304789
Percent: 8%
-----
Total: 3940124

-> Final RRI: 8.7507

-> Final Categories (66): [servico filmes modelagem lojas estrutura servicos
  financeiros combustivel localizacao entreterimento caixa automotivo limpeza
  amenities telefones e celulares delivery eventos wi-fi aulas acessorios lu
  ar experiencia recepcao governanca conveniencia qualidade produtos lazer gra
  duacao audio organizacao custo beneficio opcoes estacionamento ingressos ca
  ro moda servico aplicativo pos graduacao eletrodomesticos moveis telefone m
  nutencao funcionamento provador chocolate pecas cinema praca de alimentacao
  atendimento alimentacao comodidades agro beleza e saude eletroportateis com
  da bebida financeiro higiene alimentos TV e video preco cardapio entrega sh
  ws bebidas]

-> Done!
go run main.go 7.14s user 0.93s system 118% cpu 6.796 total
```







**“Talk is  
cheap. Show  
me the code.”**

**Linus Torvalds**

# Conclusion

# Links

- *Introdução ao Node.js (Single-Thread, Event-Loop e mercado)*
- *Is Node.js Single-Threaded or Multi-Threaded? and Why?*
- *Parallel processing in Node.js using worker threads*
- *Piscina NPM package*
- *Avoiding Memory Leaks in Node.js: Best Practices for Performance*
- *Usando worker\_threads em Node.js*
- *Not everything goes to worker threads in Node.js*

# Links

- [#11 Meetup \(online\) da comunidade Go de SC](#)
- [Generics - Um guia para o engenheiro ansioso](#)
- [Identificando pontos de melhorias com o Go Profiling](#)
- [gRPC para quem só conhece REST](#)
- [14 Meetup \(online\) da comunidade Go de SC](#)
- [O que são e como funcionam as Goroutines](#)
- [Goroutines e go channels](#)
- [Introdução a Go - Elton Minetto & Leandro Lugaresi](#)
- [Golang concurrency](#)

# Links

- *avelino/awesome-go*
- *A semana Go (newsletter)*
- *A Linguagem de Programação Go (book)*

# Perguntas? :)

**Leonardo Rifeli**

*Co-founder & CTO*

leonardo.rifeli@harmo.me

**harmo.me**

**HARMO**

Uma empresa **ReclameAQUI**