Estudo sobre a Linguagem de Programação Go

Breno Matheus de Santana Veloso e Danielle Assis Rufino Borges

[1]Departamento de Ciência da Computação – Universidade Federal de Roraima (UFRR)
Boa Vista – RR – Brazil

bremathx@gmail.com, danielleassisr@gmail.com

Go, ou Golang, é uma linguagem ainda nova no mercado da computação por possuir apenas 5 anos desde a sua primeira versão completa, mas que vem ganhando espaço de maneira rápida por possuir características altamente atrativas e que que vêm se tornando cada vez mais necessárias ao sistemas de computação. Por ser voltada ao paradigma de programação concorrente, ela consegue alcançar um tempo de resposta muito baixo, além de que tem parte de sua estruturação semelhante ao já aclamado C, mas sem deixar de ser uma linguagem de fácil aprendizagem.

1. História do surgimento da linguagem

GO, em seu início, era desenvolvida como um projeto de tempo parcial por engenheiros da gigante de buscas Google no ano de 2007. Foi fundada por Ken Thompson, que participou de vários projetos bem sucedidos, entre eles a linguagem B, C, Unix e UTF-8; Rob Pike, que também participou de projetos como Unix e UTF-8; e Robert Griesemer, que participou de projetos como Hotspot e JVM, além de alguns outros engenheiros da Google. Um ano após sua fundação, passou de um projeto de tempo parcial para um projeto de tempo integral da própria Google, em 2008, sendo que, em 2009, a o projeto se tornou open source. A sua primeira versão só foi lançada em 2012, com as versões 1.1 e 1.2 sendo lançadas em 2013 em 2014, a 1.3 e 1.4.

2. Domínios de aplicação

Uber - os desenvolvedores do Uber disseram que os seus algoritmos de geo-referência (o que é muito utilizado no aplicativo) muitas vezes apresentavam funcionalidades duplicadas. Com o Go, além de não exigir um grande grau de experiência dos desenvolvedores, a linguagem ainda apresentou tempos de respostas mínimos em seus algoritmos.

Dropbox - foi migrado da linguagem Python, a qual os desenvolvedores se sentem muito agradecidos pelo que ela proporcionou, para a linguagem Go por ela lidar melhor com concorrência (de processos) e por seu rápido tempo de execução. A publicação sobre a mudança segue nas referências e as bibliotecas open source de Go do Dropbox podem ser encontradas no repositório https://github.com/dropbox/godropbox.

3. Paradigmas suportados pela linguagem

Programação Concorrente:

```
import "fmt"


func f(from string) {
   for i := 0; i < 3; i++ {
       fmt.Println(from, ":", i)
   }
}


func main() {


  f("direct")


  go f("goroutine")


  go func(msg string) {
       fmt.Println(msg)
   }("going")


  var input string
   fmt.Scanln(&input)
   fmt.Println("done")
}
```

Orientado a Objetos:

```
type Creature struct {

 Name string
```

```
 Real bool

}
func (c Creature) Dump() {
 fmt.Printf("Name: '%s', Real: %t\n", c.Name, c.Real)
}
type FlyingCreature struct {
 Creature
 WingSpan int
}
dragon := &FlyingCreature{
   Creature{"Dragon", false, },
   15,
}
fmt.Println(dragon.Name)
fmt.Println(dragon.Real)
fmt.Println(dragon.WingSpan)
type Fooer interface {
 Foo1()
 Foo2()
 Foo3()
}
type Foo struct {
}
func (f Foo) Foo1() {
   fmt.Println("Foo1() here")
}
func (f Foo) Foo2() {
   fmt.Println("Foo2() here")
}
func (f Foo) Foo3() {
   fmt.Println("Foo3() here")
}
type foo struct {
}
func (f foo) Foo1() {
   fmt.Println("Foo1() here")
}
func (f foo) Foo2() {
   fmt.Println("Foo2() here")
}
func (f foo) Foo3() {
   fmt.Println("Foo3() here")
}
func NewFoo() Fooer {
   return &Foo{}
```

```
}
f := NewFoo()
f.Foo1()
f.Foo2()
f.Foo3()<br>
type SuperFooer struct {
 Fooer
}
func main() {
 s := SuperFooer{}
 s.Foo2()
```

4. Variáveis e tipos de dados

Os tipos de dados suportados pelo Go, e seus respectivos tamanhos em bytes, são:

TIPO	TAMANHO	TIPO	TAMANHO
bool	1	uint32	4
string	16	uint64	8
int	8	uintptr	8
int8	1	byte	1
int16	2	rune	4
int32	4	float32	4
int64	8	float64	8
uint	8	complex64	8
uint8	1	complex128	16
uint16	2		

5. Comandos de controle

For:

```
import "fmt"


func main() {
```

```
 i := 1
  for i <= 3 {
      fmt.Println(i)
      i = i + 1
   }


  for j := 7; j <= 9; j++ {
      fmt.Println(j)
   }


  for {
      fmt.Println("loop")
      break
   }
}
```

If/Else:

```
import "fmt"


func main() {
  if 7%2 == 0 {
       fmt.Println("7 é par")
   } else {
       fmt.Println("7 é ímpar")
   }


 if 8%4 == 0 {
       fmt.Println("8 é divisível por 4")
   }


  if num := 9; num < 0 {
```

```
        fmt.Println(num, "é negativo")
    } else if num < 10 {
        fmt.Println(num, "tem 1 dígito")
    } else {
        fmt.Println(num, "tem vários dígitos")
    }
}
```

Switch:

```
import "fmt"
import "time"


func main() {


i := 2
    fmt.Print("escreva ", i, " como ")
    switch i {
    case 1:
        fmt.Println("um")
    case 2:
        fmt.Println("dois")
    case 3:
        fmt.Println("três")
    }
  switch time.Now().Weekday() {
    case time.Saturday, time.Sunday:
        fmt.Println("é final de semana")
    default:
        fmt.Println("é dia de semana")
    }
  t := time.Now()
```

```
  switch {
  case t.Hour() < 12:
      fmt.Println("é antes do meio dia")
  default:
      fmt.Println("é depois do meio dia")
  }
}
```

6. Escopo (regras de visibilidade)

Go aceita variáveis em pacotes ou a nível de função, sendo que, dentro da função, pode-se usar atribuição rápida com tipo de variável implícito. O exemplo a seguir mostra os dois escopos de uso das variáveis:

```
package main

import "fmt"

var c, python, java bool // Variáveis declaradas em pacote;

func main() {

var i int // Variável declarada a nível de função

fmt.Println(i, c, python, java)

}
```

7. Exemplo prático de uso da linguagem de programação

A proposta do exemplo prático é criar um script que a cada uma certa quantidade de tempo determinada, envie um e-mail usando a conta do usuário para um outro endereço de e-mail. Como paradigma, foi escolhido o de Programação Concorrente, pois a linguagem estudada tem um foco muito grande neste tipo de paradigma e, para trazer um exemplo dele, mesmo que simples, trazemos um script que, ao mesmo tempo em que contabiliza o tempo necessário, ele executa a tarefa sem perder o tempo da execução durante a contabilização.

8. Conclusões

Como GO ainda é uma linguagem emergente, não se vê o seu uso frequente entre aplicações famosas já estabelecidas dentro do cenário tecnológico atual, mas que aos poucos já está ganhando espaço, como no caso do Uber, um aplicativo de sucesso, usar a mesma por conta do tempo de resposta reduzido, e como no Dropbox, que migrou de Python ao GO por conta de seu foco em programação concorrente, que, além de reduzir o tempo de execução, torna a realização de funções menos propensas a duplicatas. Apesar de ser uma linguagem nova, é uma linguagem de fácil aprendizado e que tem um potencial muito grande, podendo vir a substituir linguagens já consagradas e antigas, como o próprio C, ao qual se assemelha.

9. References

REVISTABW. Go Lang: Conceitos Iniciais.Revista Brasileira de Web: Tecnologia. Disponível em http://www.revistabw.com.br/revistabw/go-lang-conceitos-iniciais/. Criado em: 04/09/2015. Última atualização: 04/09/2015. Acessado em 15/04/2017.

Wei, Kai. “HOW WE BUILT UBER ENGINEERING’S HIGHEST QUERY PER SECOND SERVICE USING GO“. Disponível em: https://eng.uber.com/go-geofence/. Criado em: 24/02/2016. Acessado em 15/04/2017.

DROPBOX, Lee, Patrick. “Open Sourcing Our Go Libraries”. Disponível em: https://blogs.dropbox.com/tech/2014/07/open-sourcing-our-go-libraries/. Criado em: 01/07/2014. Acessado em 15/04/2017.

https://go-tour-br.appspot.com. Acessado em:17/07/2017