Урок Спринт 2.3

Race Detector

Race Detector (детектор гонок) в Go — это инструмент, предоставляемый компилятором Go для выявления гонок данных в параллельных программах. Гонки данных возникают, когда несколько горутин обращаются к общим данным без синхронизации, что может привести к неопределённому поведению программы.

`Race Detector` Go (иногда называемый "`go race`") позволяет обнаруживать гонки данных, а именно состояния, в которых одна горутина пытается читать или писать данные, которые в то же время изменяются другой горутиной без правильной синхронизации.

Чтобы запустить `Race Detector` в Go, используется флаг `-race` при вызове команды `go run`, `go build` или `go test`. Например:

```
go run -race имя_файла.go
```

После запуска программы или тестов с флагом `-race`, компилятор Go отслеживает доступ к общим данным из разных горутин и выводит предупреждения в случае обнаружения гонок. Эти предупреждения обычно содержат информацию о месте в коде, где произошла гонка, и о переменных, вызвавших гонку.

Рассмотрим код, триггерящий `race detector`.

```
package main

import (
    "fmt"
    "sync"
)

func main() {
    var wg sync.WaitGroup
    var sharedData int
```

```
    iterations := 100

    for i := 0; i < iterations; i++ {
        wg.Add(1)
        go func() {
            sharedData++ // Несинхронизированный доступ к общим данным
            wg.Done()
        }()
    }

    wg.Wait()
    fmt.Println("Final value of sharedData:", sharedData)
}
```

Запустим race detector и глянем, что он нам показывает:

```
WARNING: DATA RACE
Read at 0x00c000094038 by goroutine 9:
  main.main.func1()
      /Users/yurasargsyan/GolandProjects/awesomeProject16/go-concurrency-exercises/main.go:16 +0

Previous write at 0x00c000094038 by goroutine 6:
  main.main.func1()
      /Users/yurasargsyan/GolandProjects/awesomeProject16/go-concurrency-exercises/main.go:16 +0

Goroutine 9 (running) created at:
  main.main()
      /Users/yurasargsyan/GolandProjects/awesomeProject16/go-concurrency-exercises/main.go:15 +0

Goroutine 6 (finished) created at:
  main.main()
      /Users/yurasargsyan/GolandProjects/awesomeProject16/go-concurrency-exercises/main.go:15 +0
==================
Final value of sharedData: 100
Found 1 data race(s)
```

`Race Detector` сообщает о гонке между чтением (`Read`) и предыдущей записью (`Previous write`) переменной `sharedData`. Это говорит о том, что несколько горутин пытаются одновременно читать и записывать общую переменную без синхронизации.

В коде создается 100 горутин, каждая из которых инкрементирует переменную `sharedData` без защиты от гонок данных (без использования мьютекса или других механизмов синхронизации). Как следствие, происходит состязание за доступ к этой переменной.

Поправим наш код и запустим еще раз:

```
package main

import (
    "fmt"
    "sync"
)
```

```go
func main() {
    var wg sync.WaitGroup
    var sharedData int
    var mu sync.Mutex // Создаем мьютекс

    iterations := 100

    for i := 0; i < iterations; i++ {
        wg.Add(1)
        go func() {
            mu.Lock() // Блокировка мьютекса перед доступом к общим данным
            sharedData++
            mu.Unlock() // Разблокировка мьютекса после завершения доступа
            wg.Done()
        }()
    }

    wg.Wait()
    fmt.Println("Final value of sharedData:", sharedData)
}
```

Последующий запуск `race detector` не выведет ничего, кроме самого результата работы приложения

```
Final value of sharedData: 100
```