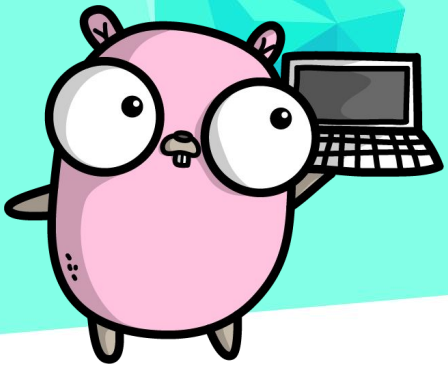
Goroutines for Speed

Erik Dubbelboer



https://github.com/erikdubbelboer

https://poki.com

```go
package main

import (
  "sync"
  "sync/atomic"
  "time"
)

func isPrime(n int) bool {
  for i := 2; i < n; i++ {
    if n%i == 0 {
      return false
    }
  }
  return true
}

func main() {
  start := time.Now()

  println(countPrimes(2_000_000, 2_100_000))

  println(time.Since(start).String())
}
```

```go
func countPrimes(from, to int) int {
  primes := 0
  for i := from; i < to; i++ {
    if isPrime(i) {
      primes++
    }
  }
  return primes
}
```

```go
func countPrimes(from, to int) int {
  primes := 0
  for i := from; i < to; i++ {
    if isPrime(i) {
      primes++
    }
  }
  return primes
}
```

```
$ go run serial.go
6872
1m33.88s
```

```go
func countPrimes(from, to int) int {
  var primes int64
  var wg sync.WaitGroup

  in := make(chan int, 16*10)

  for i := 0; i < 16; i++ { // Start workers.
    wg.Add(1)
    go func() {
      for i := range in {
        if isPrime(i) {
          atomic.AddInt64(&primes, 1)
        }
      }

      wg.Done()
    }()
  }

  for i := from; i < to; i++ { // feed numbers to workers.
    in <- i
  }

  close(in) // Make sure workers terminate.
  wg.Wait() // Wait for workers to terminate.

  return int(primes)
}
```

```go
func countPrimes(from, to int) int {
  var primes int64
  var wg sync.WaitGroup

  in := make(chan int, 16*10)

  for i := 0; i < 16; i++ { // Start workers.
    wg.Add(1)
    go func() {
      for i := range in {
        if isPrime(i) {
          atomic.AddInt64(&primes, 1)
        }
      }

      wg.Done()
    }()
  }

  for i := from; i < to; i++ { // feed numbers to workers.
    in <- i
  }

  close(in) // Make sure workers terminate.
  wg.Wait() // Wait for workers to terminate.

  return int(primes)
}
```

```
$ go run workers.go
6872
9.18s
```

```go
func countPrimes(from, to int) int {
  var primes int64
  var wg sync.WaitGroup

  for i := from; i < to; i++ {
    wg.Add(1)
    go func(i int) {
      if isPrime(i) {
        atomic.AddInt64(&primes, 1)
      }

      wg.Done()
    }(i)
  }

  wg.Wait() // Wait for all goroutines to finish.

  return int(primes)
}
```

```go
func countPrimes(from, to int) int {
  var primes int64
  var wg sync.WaitGroup

  for i := from; i < to; i++ {
    wg.Add(1)
    go func(i int) {
      if isPrime(i) {
        atomic.AddInt64(&primes, 1)
      }

      wg.Done()
    }(i)
  }

  wg.Wait() // Wait for all goroutines to finish.

  return int(primes)
}
```

```
$ go run parallel.go
6872
9.84s
```

Questions?

Gopher by Ashley McNamara