Passwortwiederherstellung in Go

Ajun Anpalakan

Agenda

1. I/O in Go
2. MD5
3. Ablauf
4. Code Implementierung
5. Performance Messung
6. Unterschied zur vorherigen Version

I/O in Go

- I/O = Input/Output
- Packages:
  - os
  - io
  - bufio
  - fmt

Zentrale Interfaces aus io-Package

```go
type Writer interface {
    Write(p []byte) (n int, err error)
}
```

```go
type Reader interface {
    Read(p []byte) (n int, err error)
}
```

I/O-Operationen in Go

Ganze Datei einlesen:

```go
data, err := os.ReadFile("datei.txt")
if err != nil { log.Fatal(err) }
fmt.Println(string(data))
```

Datei zeilenweise lesen:

```go
file, _ := os.Open("datei.txt")
defer file.Close()
scanner := bufio.NewScanner(file)
for scanner.Scan() {
    fmt.Println(scanner.Text())
}
```

I/O-Operationen in Go

```go
file, err := os.Create("output.txt")
    if err != nil {
        fmt.Println("Fehler beim Erstellen:", err)
        return
    }
    defer file.Close()
    file.WriteString("Hello, World!\n")
```

Parallelisierung von I/O in Go

```go
package main

import (
    "os"
    "sync"
)

func main() {
    var wg sync.WaitGroup

    wg.Add(2)

    go func() {
        defer wg.Done()
        os.WriteFile("a.txt", []byte("Hallo A"), 0644)
    }()

    go func() {
        defer wg.Done()
        os.WriteFile("b.txt", []byte("Hallo B"), 0644)
    }()

    wg.Wait()
}

```

Was ist MD5?

- Message Digest Algorithm 5
- Akzeptiert Nachrichten beliebiger Länge als Eingabe
- crypto/md5 library
- Beispiel:

md5("password1") = 7c6a180b36896a0a8c02787eeafb0e4c

Ablauf

- Datei rockyou.txt einlesen
- Transformieren der Wörter (Sonderzeichen & Ziffern)
- MD5-Hash jedes transformierten Wortes berechnen
- Vergleich mit den gegebenen Hashes
- Treffer → Ausgabe & Abbruch der Suche

Ansatz

- Datei wird in mehreren Chunks aufgeteilt
  - Jeder Chunk endet an einem Zeilenumbruch
- mehrere Goroutines verarbeiten die Chunks gleichzeitig
- Atomare Operationen
- Bitmaske für gefundene Hashes

Code Implementierung

Performance Messung auf SAP MacBook

```
I750480@L297QW5X6Q password % ./hashcrack_macarm
8
MATCH: "Meatloaf9" => Hash d31daf6579548a2a1bf5a9bd57b5bb89
MATCH: "123mango#" => Hash 648d5d9cc7cafe536fdbc6331f00c6a0
MATCH: "+ ann1792" => Hash 32c5c26e20908ebd80269d32f51cb5bb
Ergebnisse:
Hash 32c5c26e20908ebd80269d32f51cb5bb => Passwort: "+ ann1792" (in 3.251805041s)
Hash 648d5d9cc7cafe536fdbc6331f00c6a0 => Passwort: "123mango#" (in 722.230791ms)
Hash d31daf6579548a2a1bf5a9bd57b5bb89 => Passwort: "Meatloaf9" (in 637.525708ms)
Gesamtdauer: 3.251922333s
Verbrauchter Speicher: 133.7166 MB
I750480@L297QW5X6Q password % ./hashcrack_macarm
8
MATCH: "Meatloaf9" => Hash d31daf6579548a2a1bf5a9bd57b5bb89
MATCH: "123mango#" => Hash 648d5d9cc7cafe536fdbc6331f00c6a0
MATCH: "+ ann1792" => Hash 32c5c26e20908ebd80269d32f51cb5bb
Ergebnisse:
Hash 32c5c26e20908ebd80269d32f51cb5bb => Passwort: "+ ann1792" (in 2.807617208s)
Hash 648d5d9cc7cafe536fdbc6331f00c6a0 => Passwort: "123mango#" (in 594.608125ms)
Hash d31daf6579548a2a1bf5a9bd57b5bb89 => Passwort: "Meatloaf9" (in 230.174166ms)
Gesamtdauer: 2.807725833s
Verbrauchter Speicher: 133.7166 MB
```

Performance Messung auf HP-Laptop

```
PS C:\Users\Anpalaka\VSC_Projekte\AllGoProjects\GithubGo\wwi24sea-konzepte-golang\password> go run .
MATCH: "+ ann1792" => Hash 32c5c26e20908ebd80269d32f51cb5bb
MATCH: "Meatloaf9" => Hash d31daf6579548a2a1bf5a9bd57b5bb89
MATCH: "123mango#" => Hash 648d5d9cc7cafe536fdbc6331f00c6a0
Ergebnisse:
Hash 32c5c26e20908ebd80269d32f51cb5bb => Passwort: "+ ann1792" (in 751.6872ms)
Hash 648d5d9cc7cafe536fdbc6331f00c6a0 => Passwort: "123mango#" (in 2.046022s)
Hash d31daf6579548a2a1bf5a9bd57b5bb89 => Passwort: "Meatloaf9" (in 907.1195ms)
Gesamtdauer: 2.047367s
Verbrauchter Speicher: 133.9087 MB
PS C:\Users\Anpalaka\VSC_Projekte\AllGoProjects\GithubGo\wwi24sea-konzepte-golang\password> go run .
MATCH: "+ ann1792" => Hash 32c5c26e20908ebd80269d32f51cb5bb
MATCH: "Meatloaf9" => Hash d31daf6579548a2a1bf5a9bd57b5bb89
MATCH: "123mango#" => Hash 648d5d9cc7cafe536fdbc6331f00c6a0
Ergebnisse:
Hash 32c5c26e20908ebd80269d32f51cb5bb => Passwort: "+ ann1792" (in 543.2368ms)
Hash 648d5d9cc7cafe536fdbc6331f00c6a0 => Passwort: "123mango#" (in 2.6015555s)
Hash d31daf6579548a2a1bf5a9bd57b5bb89 => Passwort: "Meatloaf9" (in 673.0498ms)
Gesamtdauer: 2.6036642s
Verbrauchter Speicher: 133.9086 MB
```

Unterschiede zur vorherigen Version

Vorherige Version

- Zeilenweise (bufio.Scanner)
- Zeilen über Channel an Worker
- 11 Sekunden

+ geringerer Speicherverbrauch
- dauert länger

Jetzige Version

- Ganze Datei (os.ReadFile)
- In Chunks gesplittet
- 2-4 Sekunden

+ sehr schneller Zugriff auf Daten
- Hoher Speicherverbrauch

Quellen

https://pkg.go.dev/std

https://medium.com/@andreiboar/fundamentals-of-i-o-in-go-c893d3714deb

https://de.wikipedia.org/wiki/Message-Digest_Algorithm_5

https://golang.cafe/blog/golang-writer-example.html

Vielen Dank für eure Aufmerksamkeit!