Paralelní programování

Robert Čížek, Lukáš Hlavatý, Peter Smatana

27. června 2017

1 Zadání projektu

Cílem projektu bylo vytvořit program který se snaží hrubou silou rozšifrovat šifrový text. Šifrovaný text je vytvořený pomocí Vigenèrovy šifry.

2 Požadavky na softwarové vybavení

Projekt je vytvořený v progamovacím jazyku Java ve verzi 1.7. Pro vytvoření stuktury projektu, pro běh a tvorbu *.jar* souborů používáme technologii Maven. Pro běh projektu je třeba mít nainstalované Java Runtime Environment. Pro práci na projektu je třeba mít Java Development Kit.

3 Popis projektu

Naše řešení je schopno text šifrovat i dešifrovat. Šifrování se provádí ve třídě `Encryptor` v metodě `Encrypt`. V této metodě iterujeme nad otevřeným textem a podle klíče jej šifrujeme. O to se stará třída `Alphabet`, která má metody `shiftUp` a `shiftDown`. Metoda `shiftUp` šifruje text podle klíče a metoda `shiftDown` dešifruje text podle klíče. Ve skutečnosti se jedná o hledání patřičného znaku ve Vigenèrově čtverci.

Ve zdrojovém kódu č. 1 je vidět jak se pracuje Vigenèrovým čtvercem. Nalezení zašifrovaného znaku je přičtení ordinální pozice znaku klíče v ASCII tabulce.

```
1 public static char shiftUp(char c, char key) {
2     int num = (int) c - 97;
3     if ((num >= 0) && (num <= 25)) {
4         num += (int) key - 97;
5         if (num > 25) {
6             num -= 26;
7         }
```

```
8           return alphabet.charAt(num);
9       } else {
10          return c;
11      }
12  }
```

Zdrojový kód č. 1: Šifrování znaku podle znaku v klíči.

Pro dešifrování máme třídu `Decryptor` ve které jsou tři metody kde uplatňujeme odlišné přístupy na dešifrování. První metoda `KeyDecrypt` dešifruje text inverzně stejně jako je implementováno šifrování. Vezme šifrový text, klíč a obdobně jako v ukázce zdrojového kódu č. 1 dešifruje text. Druhý přistup reprezentuje metoda `FreqDecrypt`, která provádí frekvenční analýzu. Frekvenční analýza je přístup, jak najít v šifrovém textu nějakou podobnost s přirozeným jazykem, ve kterém je zpráva napsaná. Poslední přístup je hledání klíče hrubou silou podle slovníku. Hádáme klíč, klíčem rozšifrováváme slovo a toto slovo hledáme ve slovníku.

4 Paralelní vylepšení programu

Paralelizaci jsme prováděli pro dešifrování textu. Šifrování jsme implementovali pro potřebu získání zašifrovaného textu.

Dešifrování je provedeno rozdělením šifrovaného textu do částí které jsou následně předány vláknům. Vlákna nezávisle na sobě zpracovávají dané části a výsledky zapisují do společné proměnné pomocí metod nebo do patřičného pole podle svého ID. Počet spuštěných vláken je závislý na délce šifrovaného textu.

```
1  public static String Encrypt(String text, String key){
2      sKey = key;
3      StringBuilder result = new StringBuilder();
4      String[] batch = Stringer.Split(text);
5      Thread[] bank = new Thread[batch.length];
6      count = new CountDownLatch(batch.length);
7      sBatch = new String[batch.length];
8      for(int i = 0; i < batch.length; i++){
9          bank[i] = new Thread(new Encryptor(batch[i], i));
10         bank[i].start();
11     }
12
13     try {
14         count.await();
15     } catch (InterruptedException ex) {
16         Logger.getLogger(
17             Encryptor.class.getName()).log(Level.SEVERE, null, ex);
```

```
18      }
19
20      for (String s : sBatch) {
21          result.append(s);
22      }
23      return result.toString();
24  }
```

Zdrojový kód č. 2: Rozdělení textu a předání vláknům.

Data **název souboru**	počet znaků	počet slov
nesmysl.txt	0	0
pohadka.txt	0	0
shakespeare.txt 0	0	

5 Měření

todo

5.1 Použitý hardware

Procesor	**Paměť**
Intel Pentium DualCore E6300	4BG RAM DDR2
Intel i5-3210	4GB RAM DDR3

6 Práce na projektu

Na projektu se všichni členové týmu podíleli stejným dílem. Projekt jsme umístili na github: `https://github.com/petersmatana/ppp`

7 Závěr

todo