

Exploration and Presentation - Assignment 3

Martin, Frederiksen
cph-mf237@cphbusiness.dk

Andreas, Vikke
cph-av105@cphbusiness.dk

Asger, Sørensen
cph-as466@cphbusiness.dk

William, Huusfeldt
cph-wh106@cphbusiness.dk

13. april 2021

Indhold

1	Opgaven	2
1.1	Opgave 1	2
2	Løsningen	3
2.1	Programmet	3

Kapitel 1

Opgaven

1.1 Opgave 1

1. Find a point in your pgrom that can be optimized (for speed), for example by using a profiler.
2. Make a measurement of the point to optimize, for example by running a number of times, and calculating the mean and standard deviation (see the paper from Sestoft).
3. If you work on the **letterfrequencies** program, make it at least 50% faster.

Kapitel 2

Løsningen

2.1 Programmet

Vi har valgt at arbejde på programmet **letterfrequencies**. Vi havde problemer med at få profileren fra intellij til at fungere korrekt og derfor har vi arbejdet ud fra det eneste hint vi fik fra intellij hvilket var at maps put funktion skulle optimeres. Vi starter lige med at angive spredningen udregnet på 10000 gennemkørsler:

56859375,2 ns \pm 10085179,175

Vi kan se et problem i forhold til maps put metode, idet put er langsom. Dette skyldes at put først skal lede mappet igennem efter et bestemt bogstav(skrevet som ASCII i vores tilfælde) og derefter tælle en op hvis bogstavet eksistere. Hvis ikke bogstavet eksistere i mappet bliver der så kastet en exception hvor denne exception bliver grebet i catch og så først der bliver bogstavet puttet ind i mappet og bliver talt 1 op sådan at det illustrere at der er 1 af et givent bogstav. På denne måde bliver maps put metode en bottleneck fordi programmet skal vente på at map får puttet et bogstav ind før programmet kan køre videre. Vores optimeret program ser derfor således ud:

```
1 public class Main {
2
3     public static void main(String[] args) throws FileNotFoundException, IOException {
4         String fileName = "C:/Users/Andreas Vikke/OneDrive/Documents/Skole/SoftwareUdvikling
5 /CPH-Business-UFO/Week13/letterfrequencies/src/main/resources/FoundationSeries.txt";
6         Reader reader = new FileReader(fileName);
7         long freq[] = new long[256];
8         tallyChars(reader, freq);
9         print_tally(freq);
10    }
11
12    public static double Run() throws FileNotFoundException, IOException {
13        String fileName = "C:/Users/Andreas Vikke/OneDrive/Documents/Skole/SoftwareUdvikling
14 /CPH-Business-UFO/Week13/letterfrequencies/src/main/resources/FoundationSeries.txt";
15        Reader reader = new FileReader(fileName);
16        long freq[] = new long[256];
17        tallyChars(reader, freq);
18        print_tally(freq);
19        return 1;
20    }
21
22    private static void tallyChars(Reader reader, long[] freq) throws IOException {
23        int b;
24        while ((b = reader.read()) != -1) {
25            freq[b]++;
26        }
27    }
28
29    private static void print_tally(long[] freq) {
```

```

28     int dist = 'a' - 'A';
29     Map<Character, Long> upperAndlower = new LinkedHashMap();
30     for (Character c = 'A'; c <= 'Z'; c++) {
31         upperAndlower.put(c, freq[c] + freq[c + dist]);
32     }
33
34     Map<Character, Long> sorted = upperAndlower
35         .entrySet()
36         .stream()
37         .sorted(Collections.reverseOrder(Map.Entry.comparingByValue()))
38         .collect(
39             toMap(Map.Entry::getKey, Map.Entry::getValue, (e1, e2) -> e2,
40                 LinkedHashMap::new));
41 }
42 }

```

Listing 2.1: Main.java

Vi kan nu tage et kig på den nye spredning som også er udregnet på 10000 gennemkørsler:

$$29067911,1 \text{ ns} \pm 282264,715$$

Vi kan se den procentvise hastighedforøgelse:

$$\frac{56859375,2 - 29067911,1}{56859375,2} \times 100 \approx 48,877\%$$

Etersom vi ikke havde mulighed for flere hints fra IntelliJ prøvede vi lige en hurtig google søgning: ”how to optimize a file reader” og fandt ud af at vi kunne optimisere koden yderligere vha. en bufferedReader. BufferedReader er hurtigere fordi den ”buffer” 16kb af filen som den så leder igennem imens den spørger harddisken om de næste 16kb af filen. På denne måde skal bufferedReader kun vente på svar fra harddisken 1 gang om det er ved de første 16kb. Vi har tilføjet bufferedReader således:

```

1     public static void main(String[] args) throws FileNotFoundException, IOException {
2         String fileName = "C:/Users/Andreas Vikke/OneDrive/Documents/Skole/SoftwareUdvikling
3         /CPH-Business-UFO/Week13/letterfrequencies/src/main/resources/FoundationSeries.txt";
4         BufferedReader reader = new BufferedReader(new FileReader(fileName), 16384);
5         long freq[] = new long[256];
6         tallyChars(reader, freq);
7         print_tally(freq);
8     }

```

Listing 2.2: NewMain.java

Vi får så følgende spredning igen udregnet på 10000 gennemkørsler:

$$19755513,7 \text{ ns} \pm 806225,216$$

Afslutningsvis ender vi med følgende hastighedforøgelse i procent:

$$\frac{56859375,2 - 19755513,7}{56859375,2} \times 100 \approx 65,255\%$$