Assignment 3
Code optimization and reporting

Mads Wulff Nielsen
Claus Kramath

16. april 2021

Indhold

1 Introduktion

Programmet, vi vil forsøge at optimere, beregner frekvensen af forekomsten af et bogstav i en tekst. Programmet er klonet fra flg. github repository:

1. `https://github.com/CPHBusinessSoftUFO/letterfrequencies`

Det oprindelige program indlæser en tekstfil, gennemløber karaktererne i denne enkeltvis og akkumulerer antallet af hver karakters forekomster løbende i et HashMap. Til sidst udskrives antallet af hver karakters forekomst; frekvensen.

2 Instruktion

Herunder er instruktioner til at køre den optimerede kode på egen maskine:

1. Vores optimerede løsning kan downloades her:
   `https://github.com/CK2800/UFO/tree/master/3`
2. Åbn projektet i IntelliJ eller tilsvarende IDE.
3. Kør Main.main() fra IDE'et, bemærk at der køres 500 loops. Vores benchmark indikerer, at det bør tage ca. 75 sek. på en i5, 2.2 GHz.
4. Observer mean og standard deviation udregningerne i konsollen. Disse udregnes for hvert 50. loop.
5. Kør nu Main i kommandoprompten ved at navigere til /target/classes og skrive flg. kommando:

   ```
   java cphbusiness.ufo.letterfrequencies.Main
   ```

6. Observer igen mean og standard deviation værdierne i konsollen.

3 Fremgangsmåde

For at identificere flaskehalse i programmet, har vi brugt Java flight recorder profileren, som findes i vores IDE; IntelliJ. Af billedet herunder fremgår det, at kørslen af programmet i sin helhed især bruger tid på

```
Main.tallyChars(Reader, Map)
```

Med identifikationen af den primære flaskehals, granskede vi den originale kode for at forstå den samt fremkomme med umiddelbare optimeringsskridt. HER BILLEDE AF DEN ORIGINALE KODE. Vores umiddelbare mistanke faldt på det faktum, at koden traverserer hele filen 1 bogstav ad gangen. Dette resulterer i mange læsninger på disken, hvorfor vi drøftede følgende løsningsforslag:

1. Indlæsning af (dele af) filen til RAM.
2. Anvendelse af alternativ reader, eventuelt med indbygget buffer.
3. Læse filen i større bidder ved brug af byte arrays.

3.1 Fravalg af forslag 1

Vi indså hurtigt at indlæsning af filen inden kørslen af Main.tallyChars ville bryde med kontrakten, idet en ændring af indlæsningstidspunkt for filen vil medføre, at indlæsningsmetoden skal køres først. Dette vil medføre ændringer i kodebasen andre steder, idet en ny metode til indlæsning af filen i RAM ville skulle kaldes først.

3.2 Anvendelse af alternativ reader

Her kommer lidt om det og nogle tal og måske en lækker graf. Kodeeks. mv.

3.3 Indlæsning af filen i bidder vha. char array

Også her kommer info og lækre grafer. Kodeeks mv.

4 Vores metoder

TBD her opstiller vi en tabel med vores egne metoder og skriver lidt om forskellene. Og så kommer der billeder af de forskellige kørsler og gerne en flammegraf.