dConc Aflevering 3 - DA6

Christian Zhuang-Qing Nielsen

201504624, christian@czn.dk

November 25, 2016

1 Introduktion

Pointen i denne opgave er at se hvorvidt notifyAll() er skidt for concurrent programmer. Som eksempel bliver Producer-Buffer-Consumer fra uge 1 brugt. Rapporten er opdelt i fem trin hvor svarene på disse følger her:

2 Trin 1

Jeg har tilføjet try/catchs der hvor bufferen kaster InterruptedExceptions.

3 Trin 2

Hvis hver producer kun laver en enkelt meddelelse og consumer, så vil wait() blive kaldt to gange for hver producer (hver gang den skal frem og tilbage). Derfor vil jeg mene at tidskompleksiteten af dette i værste tilfælde er $2n(n-1)/2 = O(n^2)$, eftersom Der højst er 2 while-loops der venter på hinanden på et givent tidspunkt. wait() udgør et problem når kommer til at blive udsat for starvation, samt det høje cpu-brug ved gentagne context-switches. Desuden er der en del tidsbrug ved at skulle vække alle trådene på én gang og så derefter sætte dem alle pånær én til at vente derefter.

4 Trin 3

Her har jeg skrevet en ny main metode som tager et integer input som så er antallet af producers den skal skabe. Consumer målen tidsforbruget ved bare at printe hver gang den får en ny streng. Se grafen til sidst, der også bekræfter $O(n^2)$ tidsforbruget.

5 Trin 4

Eftersom at trådenene ikke har brug for at blive vækket i en bestemt rækkefølge, så er det helt fint bare at vække dem alle på én gang med notifyAll(). Når vi bare bruger notify() er det sværere at få vækket de relevante tråde.

6 Trin 5

Hvis vi havde fået givet en magisk notify() af Julemanden, som kun vækkede relevante tråde op hver gang, kunne vi spare hele trinnet med at vække en masse threads og så sætte dem til at sove igen med det samme. På den måde kunne vi få tiden reduceret til lineært $O(n)$ frem for polynomisk.

7 Appendix

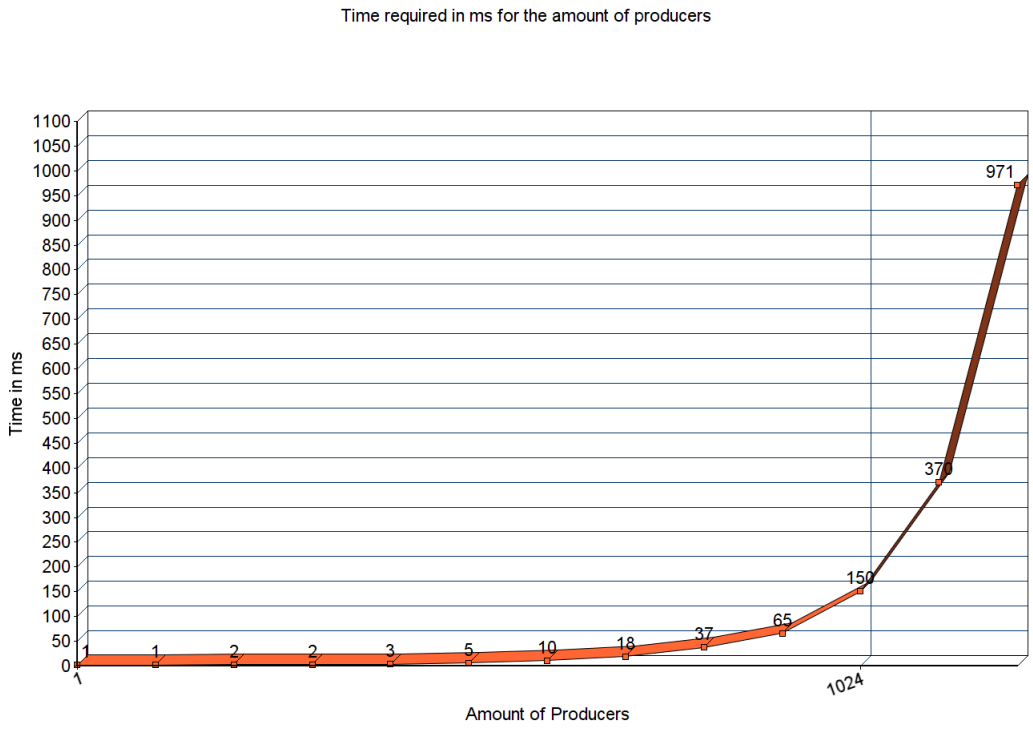

Figure 1: Tidsforbruget