Systemnära programmering 5DV088HT19

Obligatorisk uppgift 4 MFIND

Bildrättigheterna tillhöra Google inc och är under CC licens.

Max Malmer(c17mmr)
Kontakt: c17mmr@cs.umu.se
Kursansvarig: Mikael Rännar, mr@cs.umu.se

2019-11-25, v.2.0

Innehållsförteckning

1. Komplettering

Denna uppgift är nu den 25:e November 2019 kompletterad. Det jag har ändrat i denna rapport är följande:

- Jag har ändrat förordet sådant att det passar min nyare implementation.
- Jag har lagt till mer resonemang kring varför programmets prestanda planar ut och vid vilket antal trådar prestandan är som bäst.
- Till detta resonemang som återfinnes på ''Testkörningar'' så har jag också lagt till en graf som representerar 100 testkörningar (Notera att resonemanget ligger på sidan efter bilden).
- Kodförändringar återfinnes i changes.txt

2. Förord

Efter jag haft tid att komplettera mitt program så är jag mycket mer nöjd med det. Nu har jag tagit mig tid att verkligen förstå semanforerna även fast dom tekniskt sätt fungerande ändå i min förra implementation. Funktionaliteten av mitt program är ungefär densamma förutom att det nu går att ange flaggor var som helst i kommandot. T.ex. så kan du lägga alla flaggor i slutet vilket är behändigt.

3. Trådsäkerhet

Programmet som det är nu bör vara trådsäkert. Jag använder mig av en kombination av mutex lås och semanforer för att garantera detta. Om programmet körs med Helgrind så får man dock output på ett misstänkt race i memcpy i printf. Detta borde dock inte vara fallet eftersom jag uppdaterat med följande :

```
#define _POSIX_C_SOURCE 200809L
#define _XOPEN_SOURCE
```

Som jag förstått så ska printf därgenom vara trådsäkert sedan kring år 2000.

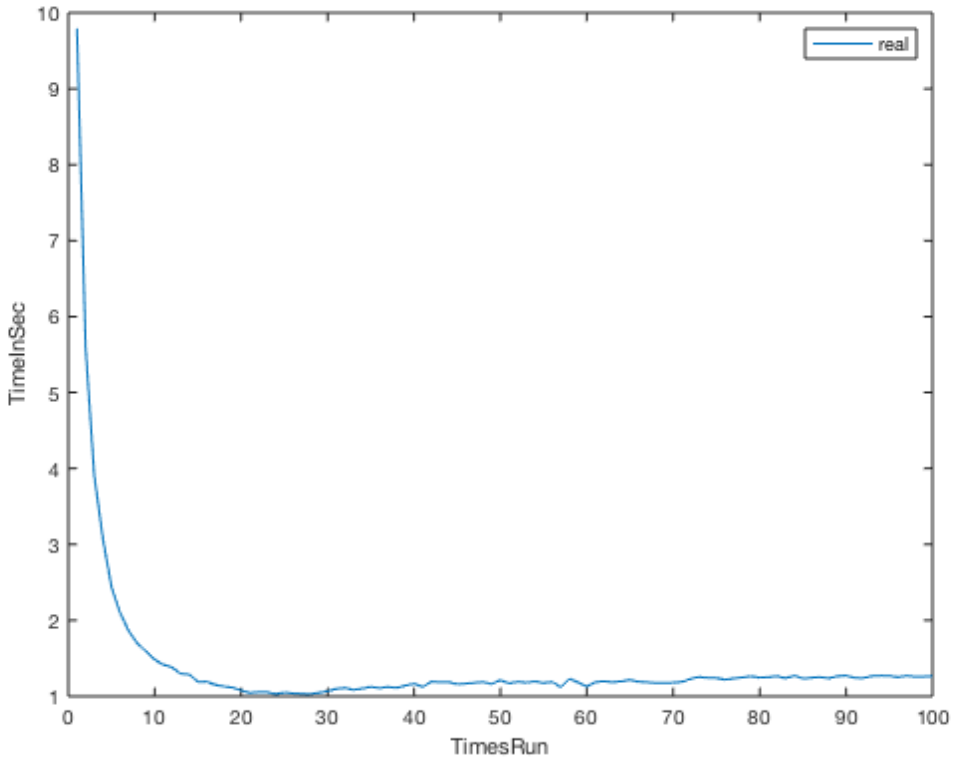
Något som dock kan vara fallet är att jag kanske låser programmet eller håller mina trådar med semanphorer lite för ofta. En tanke är t.ex. att jag låser upp semanphorer både när en ny path läggs i kön, men också när en tråd lämnar trådfunktionen. I min pseudokod så tänkte jag att det skulle vara möjligt att inte behöva låsa upp semanphoren där men så verkar inte vara fallet. Så jag misstänker att det kan vara något som jag glömt. Dock så verkar detta inte ställa till problem eftersom arbetsbelastningen (utan valgrind) alltid är jämn.

4. Arbetsfördelning mellan trådarna

Min tanke med arbetsfördelningen är att alla trådar kör samma kod(samma funktioner) och stannar vid delade resurser och väntar (t.ex. kön) med ett ’’mutex_lock’’. Sedan för att inte dom ska köra på och försöka ta av kön när den är tom så har jag både en if sats som förhindrar att detta ska vara möjligt i kombination med en semanfhor som kontrollerar så att ingen tråd kör på och skapar massor med konstiga minnesfel. Eftersom dom arbetar med samma sak allihopa så blir det lätt för dom att dela på arbetet desto fler dom är. Min ursprungliga tanke var att ha olika trådar som gör olika uppdelade arbetsuppgifter, jag tyckte dock i slutändan att detta enbart vart besvärligt när det skulle vara möjligt att köra koden för ett godtyckligt antal trådar.

5. Testkörningar

Jag gjorde ett bash script för att köra tester på koden som jag sen plottade 100 körning av gentemot tid i matlab på itchy. Som grund för datan så körde jag 10 testkörningar med 1 till 100 trådar. Programmet verkar fungera utmärkt för detta och man ser då detta plottade sammanhang:



På dessa 100 testkörningar så kan vi tydligt se till en början att fler trådar helt klart är bättre än färre. Dock så kan man se att programmet ter sig på denna maskin vara optimerat kring 25 körningar. Jag körde en testkörning på min mac som heter ''mactest.txt''. Där ser man att denna punkt inte är samma på olika maskiner, på macen till skillnad från itchy så når vi den optimala punkten redan vid 10 trådar. Det i samband med inläsning får mig att tänka att det generellt sätt alltid till en början är bättre med fler trådar men att man vid en punkt når ett ställe där fler trådar helt enkelt inte hanteras snabbare utav CPU utan att det istället tar mer tid att hantera trådarna(distibution och start) sådant att det sakta börjar ta mer tid. Att det efter 25 körningar tar mer tid är tydligt utav grafen, det går dock ganska sakta uppåt men ändock uppåt. Med andra ord så på just itchy så är det optimalt att köra 25 trådar med min mfind, men generellt sätt så är alltid fler trådar bättre så länge maskinen man arbetar på kan hantera det.

6. Referenser

1. http://man7.org/linux/man-pages/man2/semop.2.html 2019-10-25
2. https://linux.die.net/man/3/pthread_mutex_lock 2019-10-25
3. https://www.cambro.umu.se/access/content/group/57250HT19-1/Laboration%204/mfind-specification.pdf 2019-10-25
4. http://man7.org/linux/man-pages/man2/semctl.2.html 2019-10-25
5. https://se.mathworks.com/help/matlab/ 2019-10-25
6. https://ss64.com/bash/time.html 2019-10-25
7. https://www.cambro.umu.se/portal/site/57250HT19-1 2019-10-25
8. Janlert, Lars-Erik., Wiberg, Torbjörn. 2000. *Datatyper och Algoritmer*. (2:a upplagan). Lund: Studentlitteratur AB.