OU3 - mdu

2022-10-19

Theodor Jonsson
ens18trn@cs.umu.se

Systemnära programmering
5DV088

1. Resonemang kring trådsäkerhet

I detta program har de använts en mutex variabel samt en conditional variabel för att hålla programmet trådsäkert. Dessa blir skickade till huvudfunktionen för trådarna genom en struct som innehåller dessa samt kön för alla uppgifter bland annat. I denna struct finns de många variabler som skulle kunna bli åtkommen av flera trådar samtidigt. Inom huvudalgoritmen blir mutexen låst innan den får åtkomst till dessa variabler sedan blir den öppnad efter den har plockat ut sin uppgift ur kön så att andra trådar kan fortsätta.

När de kommer ett tillfälle där flera trådar skulle kunna nå samma int variabel som håller summering av filerna i ett directory blir mutexen innan för att förhindra race conditions se Figur 1.

```
void executeTask(struct task *task, Directory *dir, TaskQueue *taskPool){
    long int size = count_file(task->file_name);
    pthread_mutex_lock(&taskPool->mutexQue);
    dir->sum += size;
    destroy_Task(task);
    pthread_mutex_unlock(&taskPool->mutexQue);
}
```

Figur 1. Funktion för att summera ihop storleken av en fil till dess katalog.

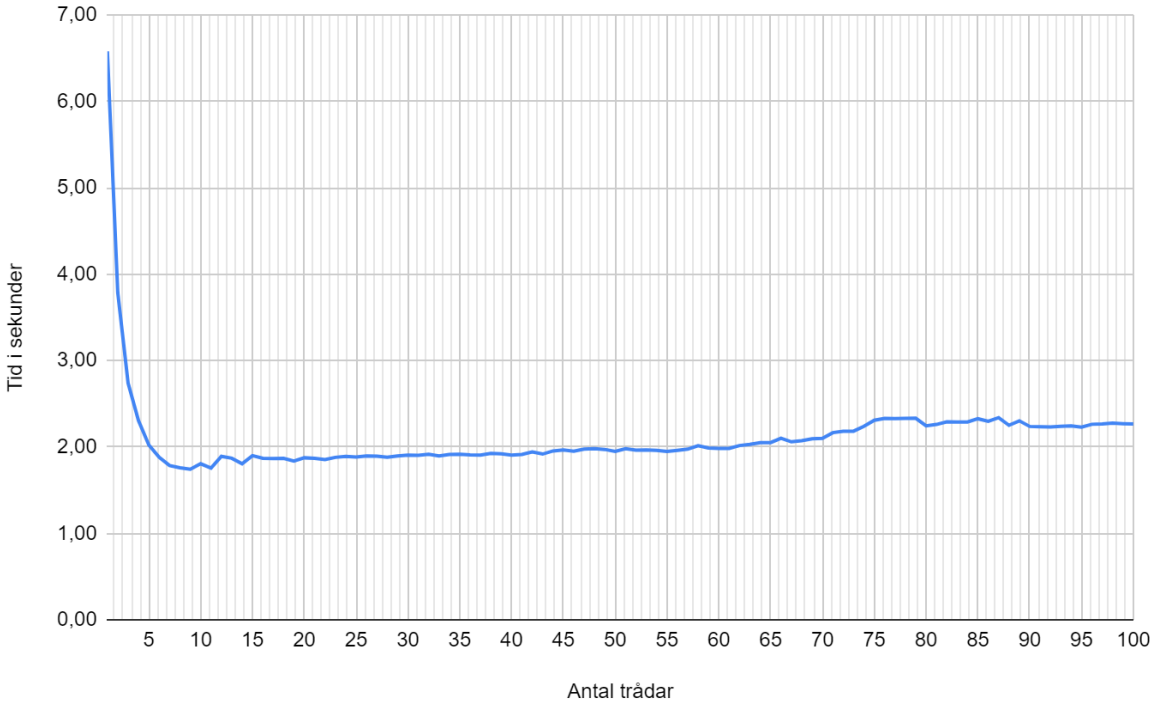
När en katalog hittas så går en tråd igenom den och summerar ihop filerna i den, om den då hittar en ny katalog så läggs denna nya katalog i kön men före den läggs in blir mutexen låst för att förhindra något fel sedan upplåst efter den har blivit inlagd se Figur 2.

```
/* If it's a new directory send it to the queue */
else if(check_if_file_or_dir(file_name) == false && strcmp(dir->d_name, "..")){
    pthread_mutex_lock(&taskPool->mutexQue);
    enqueue_Task(taskDir->dirID, file_name, taskPool);
    taskPool->finished = false;
    pthread_cond_broadcast(&taskPool->condQue);
    pthread_mutex_unlock(&taskPool->mutexQue);
}
```

Figur 2. När en ny katalog hittas och ska läggas in i kän

Theodor Jonsson
ens18trn@cs.umu.se

2. Analys över prestanda



Figur 2. Graf över exekveringstid med trådantal 1-100

I Figur 2 ser vi att körtiden sänks drastiskt fram tills ungefär 10 trådar där efter trendar den väldigt sakta uppåt tills den nått 100. Detta anser jag är på grund av att programmets algoritm för att hantera kataloger är ganska bristande.

När en tråd hittar en katalog går den igenom katalogen och summerar ihop alla filer bara om en ny katalog hittas skickas katalogen tillbaka i kön och låter en annan tråda plocka upp den. Detta verkar leda till att när de blir fler än 10 trådar väntar flera trådar på att få en ny uppgift medans de trådar som har fått sig en uppgift går igenom en massa filer.

3. Diskussion och reflektion

Största problemet för mig under denna laboration var att få till algoritmen för att gå igenom kataloger och vad som skulle skickas tillbaka för att kunna köras av en annan tråd.

Samt att i slutet av laborationen så bestämde jag mig för att byta från en vanlig array till en kö för att hålla kolla på alla uppgifter. På grund av att arrayen gav mig många fel där den försöker få tag i ett värde out of bounds. Detta är varför jag har lite missledande benämningar på vissa structs jag hade inte tid att ändra alla namn.

Jag hade velat optimera algoritmen mycket mer till exempel nu när en tråd hittar en katalog så går den igenom den och räknar ihop alla filer och skickar tillbaka till kön om den hittar en ny katalog. Men detta kan då bli att en tråd sitter och räknar ihop flera filer medans de andra trådarna waitar. Jag hade velat göra så att varje fil blir skickad tillbaka till kön så att flera trådar kan sitta och räkna ihop.

Rapporten blir nog inte så bra blev lite stressigt i slutet att försöka få klar koden.

Theodor Jonsson
ens18trn@cs.umu.se