Blekinge tekniska högskola
HT 2016
2016-09-15

handledare:
Cuong Phung
Erik Bergenholtz

Laboration 3

Minnesallokering

Adam Georgsson
Therese Karlsson

Innehållsförteckning

Inledning

I den här laborationen undersöks organisationen av minnet, som till exempel vad som inträffar när primärminnet tar slut och swapminnet tar vid. Vi undersöker bland annat hur skrivhastigheten ändras och hur mycket långsammare det går när primärminnet tar slut. Är verkligen hårddisken så mycket långsammare än RAM-minnet? Detta är en fråga som laborationen förhoppningsvis kan ge svar på.

Metod

Med hjälp av ett hjälpprogram (se bilaga 1) allokerade vi minne i block om 100 MB som fylldes med data och sedan av-allokerades. En tidsstämpel hämtades före allokeringen och en annan efteråt för att jämföra hur lång tid som har förflutit. I efterföljande rundor ökades minnes-allokeringen med 100 MB i taget fram tills dess att minnet tog slut. Minnesblocken valde vi att fylla med datatypen **char** med hjälp av funktionen **memset(void *str, int c, size_t n).** Operativsystemets sidstorlek var 4 KB men det var inget vi behövde ta hänsyn till eftersom en **char** är 1 byte och kommer alltid fylla en sida. Hela blocket kommer således också att fyllas helt och hållet. Laborationen utfördes på en dator med Debian som operativsystem med 16 GB internminne och 1 GB swapminne i en SSD. För att ta reda på informationen om minnet använde tittade vi i filen **/proc/meminfo** och sidstorleken fick vi fram genom **getconf PAGE_SIZE.**

Resultat

Resultatet av den uppmätta tiden tillsammans med hur stort datablocket var, är illustrerat av diagrammet nedan. Y-axeln består av millisekunder/MB och X-axeln beskriver antal MB. I början gick vi igenom minnesblock från 100 MB upp till cirka 16000 MB men kurvan var linjär och oföränderlig fram tills minnet började ta slut därav ändrades startvärdet till 15000 MB där det intressanta börjar ta vid. När minnet börjar ta slut kan man utläsa att allokeringen tar längre och längre tid och med hjälp av operativsystemets inbyggda systemövervakaren kunde vi se att det skedde när swapminnet fick börja hjälpa till.

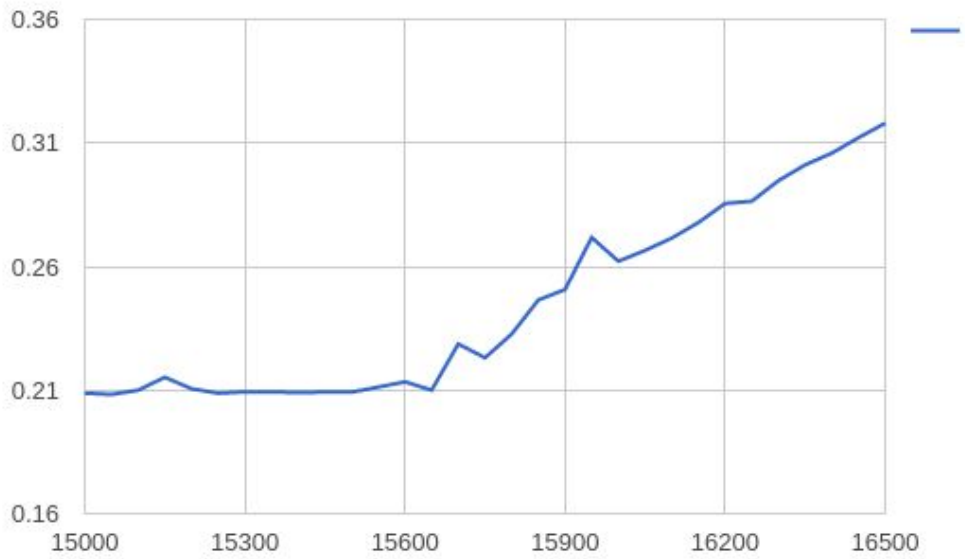

Diagram över förhållandet tiden det tar att skriva data per MB

Diskussion

Labbmiljön var ej optimal för det som skulle undersökas då minnets storlek medförde långa väntetider. När allokeringen nådde den kritiska punkten vid minnets begränsningar och swapminnet skulle ta över fanns det många gånger inte tillräckligt med swapminne för att få fram ett intressant resultat. Primärminnet var alltså alldeles för stort och swapminnet var alldeles för litet. Problemet med primärminnets storlek var lätt att lösa genom att öka startvärdet till 15000 MB. Att tömma swapminnet visade sig vara svårare eftersom vi saknade root-rättigheter. Enda sättet vi kom på för att lösa det var att starta om datorn vilket medförde viss frustration. Hade man använt en **for**-loop med en **int** som räknare så hade den inte räckt till eftersom 16 miljarder bytes skulle loopas igenom. Det är för stora tal för en **int** att hantera men vi slapp det problemet eftersom vi använde oss av **memset()**-funktionen som tar emot en **size_t** som räknare.

Anledningen till att swapminnet är långsammare är att det ligger på hårddisken som har mycket sämre läs- och skrivhastighet än primärminnet. Till en början fick vi **segmentation fault** vilket berodde på att **memset()**-funktionen var placerad utanför kontrollen om **malloc()** kunnat allokera något minne. Det var även problem med att swapminnet blev fullt och inte av-allokerades eller av-allokerades väldigt långsamt. Det löste vi genom att starta om datorn.

Referenslista

Bilaga 1, main.c