# 운영체제 2차 과제 보고서

2019320054 컴퓨터학과 고영인

Freeday 사용일수: 2

1. 과제 개요

   리눅스의 cpu schedueling 기법 중 하나인 Round Robin의 동작 과정 중 time slice가 미치는 영향대해 관찰합니다. 이를 위해 c언어를 이용한 프로그램을 작성하고, 커널을 수정하여 실험을 진행할 예정입니다. 스케쥴링에 따른 프로세스의 성능 차이를 관측하고, 이를 이용해 time slice가 cpu 성능에 미치는 영향을 분석합니다.

2. RR에서 time slice가 미치는 영향

   Round Robin기법은 cpu의 실행 과정을 동일 시간으로 쪼개서, 프로세스에 배분하는 선점형 스케줄링 방식입니다. 여기서 time slice는 각 프로세스가 cpu를 1번 점유하는 시간으로 본 실험에서는 주요한 변수입니다.

   time slice가 작을수록, cpu 할당시간이 프로세스들에 공평하게 분배됩니다. 또한, 응답성도 증가할 수 있습니다. 프로세스는 더욱 빨리 반응할 수 있으며, 작업 큐에 있는 프로세스는 짧은 시간동안 cpu를 할당받고 실행될 수 있습니다.

   그러나 time slice가 작으면 생기는 성능이 저하되는 영향도 있습니다. time slice가 작다는 것은 cpu가 프로세스를 자주 바꿔야 한다는 것이고, 이는 context switching이 자주 일어난다는 것입니다. 프로세스를 save하고 load를 더욱 많이 하게 되고, 이는 성능 저하로 연관되는 원인 중 하나입니다.

   따라서, time slice는 Round Robin 스케줄링에서 다양한 요소에 영향을 미치므로, 응답성, 공정성, context switching 등을 고려해서 적절한 크기를 정해야 합니다.

3. 소스코드 설명

   cpu.c부터 실행됐을 때부터 설명을 하겠습니다. 처음에 인자로 들어오는 개수가 3개가 아니면 곧바로 종료되게 하였습니다. 그다음, cpu 스케줄링 기법을 변경하게 하기 위해 만든 구조체 sched_attr을 이용해 변수 attr을 선언해주고, string.h에 선언되어 있는 memset함수를 이요해서, 할당을 해줍니다. 이때 사용한 구조체 sched_attr은 구조체의 크기 size, 정책을 위한 shced_policy, 스케줄링 동작 방식을 제어하는 sched_flag, sched_policy를 sched_batch, sched_other로 설정할 때, 설정할 sched_nice, sched_policy를 FIFO나 RR로 설정할 때 설정할 우선순위인 sched_priority, deadline을 위한 sched_runtime, sched_deadline, sched_period로 이루어져 있습니다. 구조체 attr에는 poliy를 이미 선언되어 있는 SCHED_RR로 설정해주고 우선순위는 10으로 설정해줍니다.

   그 후에는 실행할 프로세스의 개수 p_num, 실행시간 t_time을 atoi를 이용해서 넣어줍니다. 그리고 현재 프로세스에 스케줄링 기법을 적용하기 위해 getpid()로 pid에 할당합니다. 그다음 시스템 콜을 호출하는 sched_setattr함수를 이용해서 SYS_sched_setattr이 잘적용되었다면 !=-1, 적용안되었으면 -1이 할당되도록 result를 만들어줍니다. 스케줄링이 제대로 적용되지 않았다면 에러메시지를 띄웁니다.

   그 이후, p_num만큼 프로세스를 만들기 위해서, 반복문을 돌려줍니다. pid=fork를 해주고 fork가 작동하지 않는다면 에러메시지를 띄우고, 종료합니다. 정상 작동했다면, 부모 프로세스에서 프로세스를 만들었음을 알리고 자식 프로세스를 실행합니다. 자식 프로세스에서는 signal.h에 선언된 ㅁsignal함수로 사용자가 ctrl+c로 현재 작동중인 프로세스를 종료하고, 돌아갈때, 작동하는 액션을 넣어줍니다. 이때 INThandler는 기존 calc함수가 종료될 때 나오는 것과 동일하게 설정하였습니다. signal 이후에는 행렬연산을 위한 calc함수에 현재 (s)인 단위를 (ms)로 바꾸기 위해 1000을 곱하고 인자로 넣어줍니다.

calc에서는 단순한 행렬연산을 반복하면서 시간을 측정하고, 현재 상태를 출력해줍니다. 시간을 측정하기 위해 time.h에 선언된 timepsec 구조체를 이용해서 단위시간 100ms를 측정하기 위한, start, end, 전체 시간을 측정하기 위한 total을 선언해줍니다. 반복문을 시작하기 전에 start, total의 시간을 구해줍니다. 그 이후에 행렬연산을 진행하고, count++한 후, 끝난 시간 end를 측정해줍니다. 실행시간은 (끝난 시간 – 시작시간)으로 구하는데 tv_sec은 (s)이고, tv_nsec은 (ns)이므로 tv_sec부분에 1000을 곱하고, tv_nsec부분에 1000000을 나눠서 (ms)로 단위를 통일합니다. 그 이후에 실행시간이 100보다 크다면 현재 상태를 출력해줍니다. 몇번째 프로세스인지와 행렬연산이 얼마나 이루어졌는지, 실행시간을 출력하고, 전체 시간을 측정해줍니다. 전체 시간이 인자로 받은 시간보다 크다면 끝났다는 것을 표시하기 위해, 몇번째 프로세스 인지, 전체 카운트 수, 전체 실행시간을 print한 후 반복문을 탈출합니다.

만약 자식프로세스가 아닌 부모프로세스라면 시그널을 무시합니다.

그리고 자식프로세스의 종료 시점까지 대기하기 위해 의미없는 while을 추가하였습니다.

cpu가 실행되면서 발생하는 cpu burst를 측정하기 위해 /usr/src/linux-4.20.11/kernel/sched/stats.h을 수정하였습니다.

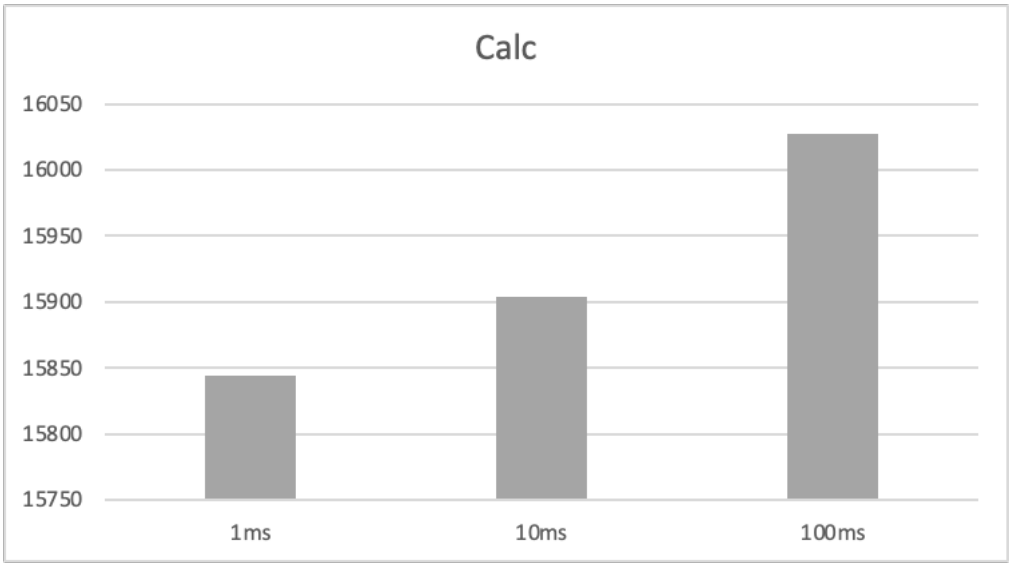
218번줄에서 현재 프로세스의 우선순위를 확인해줍니다. 이때 우선순위가 10이라는 것은 cpu.c에서 설정한 프로세스의 우선순위와 동일하다는 것이고 이때, 프로세스의 아이디와 cpu burst time(ns), 우선순위를 출력하기 위해 printk함수를 이용해서 출력을 해줍니다.

4. time slice에 따른 성능변화

각 time slice별로 5번 반복하여 평균값을 이용해서 행렬연산의 횟수를 측정하였습니다. 각각 결과는 많은 차이는 나지 않지만, 어느정도 차이를 보이고 있습니다. 이를 보면 time slice가 작을 때, context switching이 기본값이었던 100ms에 비해 10배 이상 수행되므로, 실제 연산에 영향을 미치는 것을 관찰할 수 있었습니다.

이때, time slice를 100ms로 설정했을 때, 10ms로 설정했을 때보다는 약 0.7%, 1ms로 설정했을 때보다는. 약 1.15%의 성능적으로 이득인 부분이 관찰되었습니다.

| RR Time Slice | 1ms | 10ms | 100ms |
|---|---|---|---|
| | # of calc | # of calc | # of calc |
| Process #0 | 7921.6 | 7954.4 | 8016.6 |
| Process #1 | 7923 | 7949.2 | 8010.8 |
| Total cals and Time | 15844.6 | 15903.6 | 16027.4 |




또한 cpu burst time을 종합해서 평균을 낸 시간과 연산의 총 횟수를 기록하였습니다. 이때, time slice를 100ms로 했을 때 1ms로 설정했을 때보다, 약 1.05%정도 성능에서의 이득이 있었고, 10ms로 설정했을 때보다, 약 0.8%정도 성능에서의 이득을 관찰할 수 있었습니다.

| RR Time Slice | 1ms | | 10ms | | 100ms | |
|---|---|---|---|---|---|---|
| | # of calc | Time(s) | # of calc | Time(s) | # of calc | Time(s) |
| Process #0 | 7921.6 | 30.03968482 | 7924.4 | 30.04984 | 8016.6 | 30.08063 |
| Process #1 | 7923 | 30.05577375 | 7921.2 | 29.91903 | 8010.8 | 30.07105 |
| Total cals and Time | 15844.6 | 60.09545857 | 15845.6 | 59.96887 | 16027.4 | 60.15169 |

| RR Time Slice | 1ms | 10ms | 100ms |
|---|---|---|---|
| | # of calc | # of calc | # of calc |
| Calculation per second | 263.6571943 | 264.2304159 | 266.4497251 |
| Baseline= 1ms | 100.0% | 100.217412% | 101.059152% |
| Baseline= 10ms | 99.783060% | 100.0% | 100.839914% |

따라서, 리눅스에서, time slice를 100ms로 설정한 것은 time slice를 짧게 함으로 오는 프로세스 분배의 공정성, 응답성에서의 장점보다, time slice가 짧았을 때, 발생하는 context switching 횟수의 증가로 오는 단점이 더욱 크기 때문이라고 할 수 있었습니다.

5. 과제 수행 시의 문제점, 해결사항

과제 수행 시, cpu.c에서 함수들을 구성할 때, 각 함수마다 라이브러리가 제각각이고, 사용방법도 잘나오지 않아서 코드를 작성할 때, 어려운 부분이 있었습니다. 이러한 부분은 man + 함수, 그리고 공식문서를 참고하여 해결하였습니다.

또한, cpu 코어설정을 하는 과정에서, sudo + 명령어를 해도 지속하여 permission denied가 뜨는 문제가 있었습니다. sudo su를 통해서 슈퍼유저모드로 진입한 후, 명령어들을 실행하니 잘 해결되었습니다.