Operating System
Assignment 2

학과: 컴퓨터학과
학번: 2019320137
이름: 황상민
제출일: 2023/06/04
Free day 사용: 3

1. 과제 개요 및 RR scheduling에서 Time slice의 영향

본 과제는 Round Robin Scheduling에서 Time slice의 변화에 따른 성능 변화를 관찰하기 위한 과제입니다. Round Robin scheduling은 각 프로세스별로 일정한 Time slice를 할당하여, 돌아가면서 Time slice만큼의 CPU를 사용한 뒤 다음 프로세스에게 넘겨주는 스케줄링 알고리즘입니다.

따라서 Time slice는 RR Scheduling의 성능을 결정하는 주요한 요소 중 하나입니다. Time slice가 크면 FCFS와 유사해지며, Context switching 횟수가 줄어들어 Context switching overhead를 최소화하여 CPU utilization을 높일 수 있으나 그만큼 응답성이 나빠진다는 단점이 있습니다. 반대로 Time slice가 매우 작으면 응답성이 좋아지지만, 그만큼 context switching이 자주 발생하여 성능이 떨어진다는 단점이 있습니다.

2. 작성한 코드 및 작업 내용 설명

1) cpu.c

먼저 RR 스케줄링을 적용하기 위해 sched_setattr 시스템 콜을 호출합니다. 스케줄링 정책을 Round Robin으로 변경하고, 추후의 CPU burst 측정을 위해 스케줄링의 priority를 10으로 설정합니다.

부모 프로세스에서 입력 받은 프로세스 수만큼 fork를 진행합니다. 부모 프로세스는 모든 자식 프로세스가 끝날 때까지 wait하도록 하고, 자식 프로세스 내에서 행렬 연산을 진행합니다. 이때 fork 직후에 전체 수행 시간을 재기 위한 timespec을 준비합니다(base).

자식 프로세스에서 본격적으로 행렬 연산을 수행합니다(run_process). 총 수행시간과 epoch당 수행시간을 따로 계산합니다(begin). 총 수행시간은 fork 이전에 재 두었던 base와 현재 시각의 차를 구하여 계산하고, epoch당 수행시간은 calc 함수 수행시마다 수행 시간을 재어 begin과의 차를 구하여 100ms마다 출력하도록 합니다. 종료 조건은 총 수행시간 기준으로 하여, 모든 프로세스가 물리적으로 실행된 시간이 입력으로 들어온 수행시간(ex: 30초)과 같아지도록 합니다.

추가로, sigint signal에 대한 Signal handler를 추가하여(signal_handler) 프로세스가 SIGINT 시그널에 의해 종료될 때 수행한 연산의 결과를 출력하고 종료할 수 있도록 하였습니다.

2) proc을 이용한 timeslice 변경

tutorial에서 명시된 파일을 이용, echo와 redirection을 이용해 timeslice를 1, 10, 100으로 각각 바꾸어 가면서 실험을 진행했습니다.

3) 최대 프로세서 사용 수 변경

tutorial에서 제공한 명령어를 활용하여 최대 코어 수를 1로 제한하여 실험을 진행하도록 했습니다.

4) sched_info_depart()

tutorial에서 제공한 코드를 활용하여 커널 메시지로 CPU burst time을 출력했습니다. 이때 priority가 10인 프로세스만 출력하도록 하여 실험하는 프로세스에 대한 정보만 출력되도록 했습니다.

5) dmesg 파싱 (parse.py)

dmesg 로그 파일을 파싱하여 프로세스별 CPU burst time의 합계를 구하였습니다. 먼저 grep 명령어와 pid를 이용해 dmesg 로그 중 분석할 프로세스의 burst time에 관한 로그만 뽑아내고, 이후 python3로 로그에서 burst time 값만 추출하여 합계를 구했습니다.

6) 쉘 스크립트 작성

1-5에 해당하는 과정을 반복 수행하기 위해 쉘 스크립트를 작성하였습니다. 쉘 스크립트는 time slice($1)와 반복 수행 횟수($2)를 입력으로 받아 원하는 time slice의 실험을 반복 수행하는 코드입니다. 쉘 스크립트 내부에서 (3) 최대 프로세서 수를 변경하여 스크립트 안에서도 프로세서 수가 잘 제한될 수 있도록 하였습니다.

```
root@bbang3-VirtualBox:/home/bbang3/os2# ./run 100 5
```

간략한 수행 과정은 다음과 같습니다.

1) ./cpu를 실행하여 표준 출력을 stdout.txt에 저장합니다.

2) stdout.txt를 파싱하여 실행한 프로세스의 pid를 알아냅니다.

3) 알아낸 pid를 바탕으로 dmesg 커널 메시지에서 실행한 프로세스의 CPU burst 관련 로그만 추출합니다. 추출하여 log_dmesg.txt에 저장합니다.

4) parse.py에 구현된 내용을 이용, log_dmesg.txt를 파싱하여 Burst time의 총 합계를 구합니다. 구한 내용을 burst.txt에 저장합니다.

3. 성능 변화 분석

1) Time slice에 따른 행렬 연산 성능 변화 분석

먼저 어플리케이션 레벨에서 측정한 실행 시간을 바탕으로 행렬 연산의 성능 변화를 분석하였습니다. 총 5회 실험을 진행하였으며, 아래 도표는 5회 실험의 평균치로 계산한 결과값입니다.

RR Time slice	1ms	10ms	100ms
	# of calc.	# of calc.	# of calc.
Process #0	2774.2	3025.4	3099.2
Process #1	2765.6	3027	3074.2
Total calc.	5539.8	6052.4	6173.4
Baseline=1ms	100	109.253	111.437
Baseline=10ms	91.5306	100	101.999

Figure 1. Time slice에 따른 행렬 연산 횟수

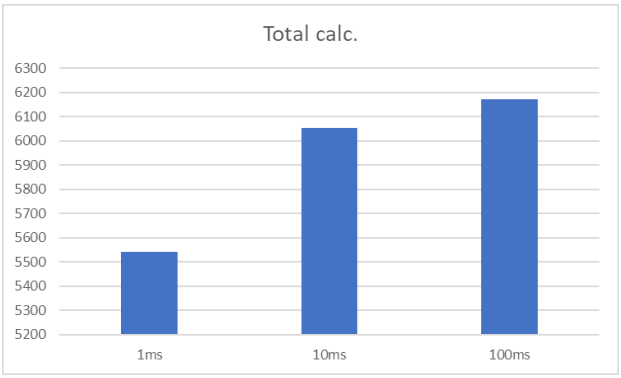

Figure 2. Time slice에 따른 행렬 연산 횟수 그래프

1ms의 경우 10ms와 비교시 약 8.5%의 연산 횟수 하락이 발생하였으며, 10ms의 경우 100ms와 비교해서 약 2%의 연산 횟수 하락이 발생하였습니다. Context switching overhead가 실제 연산 횟수에 영향을 끼친 것으로 보이며, Time slice가 줄어들수록 Context switching이 더 자주 발생하여 전체 수행시간에서 실제 연산을 수행하는 시간이 줄어든 것으로 추측됩니다.

2) CPU Burst time을 활용한 성능 변화 분석

정밀한 실험을 위해 CPU burst time을 바탕으로 단위 시간당 행렬 연산량을 분석하였습니다. 총 5회 실험을 진행하였으며, 아래 도표는 5회 실험의 평균치로 계산한 결과값입니다.

RR Time slice	1ms	10ms	100ms
Calculation per second	181.1951	187.6178	194.5215
Baseline=1ms	100	103.5446	107.3547
Baseline=10ms	96.57672	100	103.6797

RR Time Slice	1ms		10ms		100ms	
	# of calc.	Time (s)	# of calc.	Time (s)	# of calc.	Time (s)
Process #0	2721.4	15.05348	2816.4	15.02825	2920.8	15.0894
Process #1	2713.6	14.94196	2812.4	14.97311	2943.4	15.05771
Total calc. and Time	5435	29.99544	5628.8	30.00136	5864.2	30.14711

Figure 3. CPU Burst 기반 성능 변화

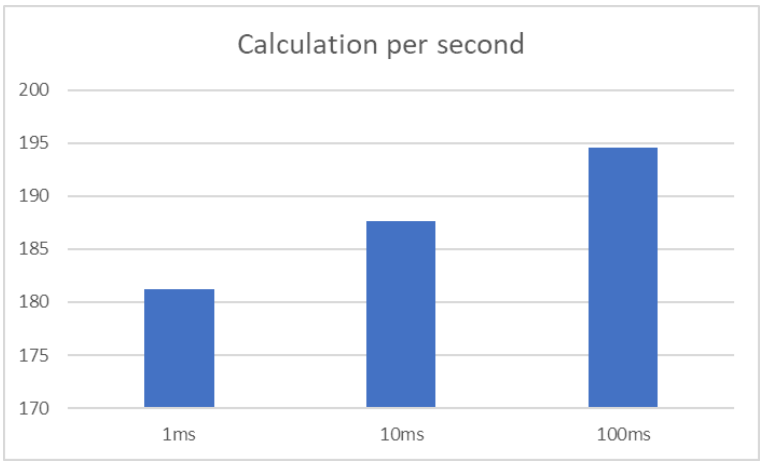

Figure 4. CPU Burst 기반 성능 변화 그래프

실험 결과 1ms의 경우 10ms와 비교시 3.5%의 성능 하락이 발생하였으며, 100ms의 경우 10ms와 비교시 3.6%의 성능 향상이 발생하였습니다. CPU Burst time 또한 Time slice에 비례해서 늘어난 것으로 보아 성능 차이의 원인은 Context switching overhead에서 기인한 것으로 보입니다. Time slice가 작을수록 Context switching이 더 자주 발생하고, 그에 따라 overhead로 인해 실제 burst time이 줄어들었고 그것이 연산 성능 하락을 불러온 것으로 추측됩니다.

4. 과제 수행 시 발생한 문제점 및 의문 사항

CPU burst time의 값이 실험 시마다 크게 변화하는 모습을 보였습니다. 경향성이 유지될 때도 있었으나, 때로는 역전되는 등 편차가 컸습니다. 구동 PC가 노트북이었는데, 전원 연결 여부나 발열 등의 문제로 CPU 성능이 매 시간마다 달라지다보니 발생한 문제인 것으로 보입니다. 결과에 반영한 실험을 진행할 때에는 최대한 다른 앱을 정리하고 발열이 없는 시점에 진행하였습니다.

의문으로 남는 것은 CPU burst time의 측정이 제대로 되고 있는 것인지에 대한 점입니다. 과제를 진행하면서 똑같은 코드를 실행했음에도 어느 PC에서 구동시키는지에 따라 burst time의 경향성이 의도한 대로 나타나기도 하고, 그렇지 않기도 하였습니다. 특히 어떤 환경에서는 최대한 모든 변인을 통제했음에도 CPU burst time이 모든 time slice에서 같게 나오는 경우도 있었습니다.

더 의문인 것은 연산 횟수의 경향성은 모든 PC 환경에서 그대로 유지(Time slice에 따라 증가)되었다는 점입니다. CPU Burst time이 모든 time slice에 대해 똑같이 나오는 PC에서는, 연산 횟수 자체의 경향성은 왜 유지되는 것인지가 궁금했습니다.