**Pintos Project 3: Threads**

담당 교수 : 박성용 교수님

학번 / 이름 : 20181683 / 전용본

개발 기간 : 2022.11.01 ~ 2022.11.06

1. **개발 목표**

* 해당 프로젝트에서 구현할 내용을 간략히 서술.

기존의 project 1,2 에서 구현한 pintos는 하나의 user program이 정상적으로 작동하는데 집중하여 개발되었고 현재 pintos scheduler는 간단한 RR(Round Robin) 방식으로 구현이 되어 있어 프로세스들 간의 priority를 고려하여 작동하고 있지 못하기에 이를 수정하여 priority 기준의 scheduler를 만드는 것이 목표이다.

1. **개발 범위 및 내용**
   1. **개발 범위**

* 아래 각 항목 개발의 필요성 또는 개발 시 기대되는 결과를 간략히 서술
  1. Alarm Clock

Timer\_sleep() 함수는 thread를 sleep 시키는데 기존에는 busy waiting 방식으로 구현되어 있어 매우 비효율적이다. run, ready state를 계속 해서 반복하므로 이를 수정하여 효율적으로 만들 수 있다.

* 1. Priority Scheduling

기존의 pintos는 RR(Round Robin) 방식으로 thread를 schedule 했다. Thread가 unblock, ready state가 될 때 priority와 상관없이 ready queue의 맨 뒤에 추가되었다.이는 thread들의 priority를 고려하지 못하기에 각 thread들의 priority를 고려하여 ready queue를 구성한다. 이를 통해 priority가 높은 thread가 먼저 schedule될 수 있게 한다.

* 1. Advanced Scheduler (추가구현을 한 경우)

Multi-level Feedback Queue(MLFQ) 혹은 Multi-level Ready Queue(MLRQ)를 이용하는 scheduler를 구현한다. 이 scheduler는 여러 개의 ready queue를 갖고 있고 각 queue의 thread들은 동일한 priority를 가진다. Scheduler에서 가장 높은 priority의 queue부터 RR(Round Robin) 방식으로 scheduler한다.

* 1. **개발 내용**
* 아래 항목의 내용만 서술

1. Blocked 상태의 스레드를 어떻게 깨울 수 있는지 서술.

Blocked 상태의 thread들을 관리하기 위한 새로운 리스트를 만든다. Thread가 sleep될 때 해당 thread를 리스트에 저장하고 해당 thread의 TCB에 block이 끝날 시간을 저장한다 이후에 매 tick마다 timer\_interrupt가 발생할 때 리스트를 순회하면서 block이 끝날 시간이 된 thread들에 대해서 unblock을 하며 block list에서 삭제한다.

1. Ready list에 running thread보다 높은 priority를 가진 thread가 들어올 경우 priority scheduling에 따르면 어떻게 해야하는지 서술.

Ready list에 running thread보다 높은 priority를 가진 thread가 들어올 경우 해당 thread가 먼저 실행되어야 한다. 현재 thread는 ready list에 들어가면서 ready list의 가장 높은 priority를 가진 thread에게 CPU를 yield해야 한다.

1. Advanced Scheduler에서 priority 계산에 필요한 각 요소를 서술. (추가구현을 한 경우)

- Nice : 각 thread는 -20 ~ 20의 값을 가지는 nice value를 갖고 있다. Nice가 양수일 경우 priority를 낮추고 0일 경우 priority에 영향을 미치지 않는다. 처음 생성된 thread의 nice 값은 0으로 초기화되며 자식 thread는 부모 thread에게 nice 값을 상속받는다. Nice 값이 변경될 때 priority도 변경되므로 재계산해야 한다.  
- recent\_cpu : 각 thread는 thread의 CPU time을 추정하는 recent\_cpu value를 갖고 있다. Recent\_cpu 값이 높을 수록 CPU time을 오래 가질 수 있다. 처음 생성된 thread의 recent\_cpu 값은 0으로 초기화되며 자식 thread는 부모 thread에게 recent\_cpu 값을 상속받는다. time interrupt가 발생할 때마다 running thread의 recent\_cpu값은 1씩 증가한다. 또한 매초마다 모든 thread의 recent\_cpu 값은 재계산된다.

Recent\_cpu = (2\*load\_avg) / (2\*load\_avg+1) \* recent\_cpu + nice

- load\_avg : load\_avg는 전역변수이며 ready state의 thread 개수 평균을 추정한다. 처음 pintos가 실행될 때 0으로 초기화되어 있으며 매초마다 재계산된다.

Load\_avg = (59/60)\*load\_avg + (1/60)\*ready\_threads

위의 요소들을 통해 priority는 4tick마다 재계산된다.

Priority = PRI\_MAX(63) – (recent\_cpu/4) – (nice\*2)

모든 요소들과 priority를 계산할 때 pintos는 floating-point 연산을 지원하지 않으므로 fixed-point 연산으로 계산한다.

1. **추진 일정 및 개발 방법**
   1. **추진 일정**

* II. A. 개발 범위를 포함하여 구현 내용에 대한 일정 작성

11.01 ~ 11.03 : Alarm clock, priority scheduling 구현

11.04 ~ 11.05 : aging, BSD scheduler 구현

11.06 : Test case 실험

11.07 : 보고서 작성

* 1. **개발 방법**
* II. B.의 개발 내용을 구현하기 위해 각각에 대해 다음 사항들을 포함하여 설명
  + 수정해야하는 소스코드
  + 수정하거나 추가해야 하는 자료구조
  + 수정하거나 추가해야 하는 함수

- Alarm clock

Devices/timer.c, timer.h 에서 먼저 block\_list를 새로 선언하며 timer\_sleep을 통해 block되는 thread들을 관리하도록 한다. thread의 TCB에 block\_time이라는 새로운 변수를 생성한다. 이를 이용해 timer\_sleep 함수에서 thread가 block될 때 thread를 block\_list에 추가하고 TCB의 block\_time에 얼마동안 block될 것인지 저장한다. 그 후에 timer\_interrupt 함수에서 time interrupt가 발생할 때마다 block\_list를 순회하며 block time이 지난 thread들을 thread\_unblock 함수를 통해 ready state로 만들어주며 block\_list에서 제거한다.

- Priority scheduling

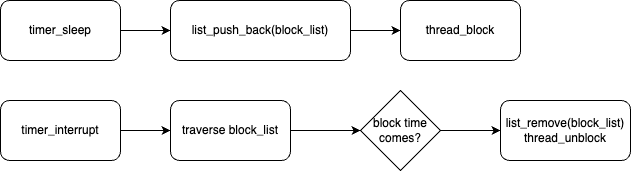
Thread/thread.c 에서 thread\_unblock, thread\_yield 함수에서 thread를 ready\_list에 저장할 때 맨 뒤에 넣지 않고 priority 내림차순으로 넣기 위해 list\_insert\_ordered 함수를 이용한다. 또 thread를 새로 생성할 때 priority가 더 높을 경우 새로 scheduling을 하기 위해 thread\_create 함수에 thread\_yield를 추가한다. 임의의 thread의 priority가 변경될 경우 현재 thread보다 priority가 높을 가능성이 있으므로 thread\_set\_priority 함수에 thread\_yield 함수를 추가한다.

Thread/synch.c 에서 sema\_down, sema\_up 함수를 수정한다. Semaphore로 인해 block되어 있는 thread들을 관리하는 waiters list를 이용한다. Sema\_down 함수에서 기존에는 waiter list에 priority에 상관없이 맨 뒤에 넣었기에 priority를 고려하여 list에 삽입하기 위해 list\_insert\_ordered 함수를 이용한다. Sema\_up 함수에서 waiter list에서 가장 높은 priority를 가진 thread를 찾아 해당 thread를 우선적으로 ready state가 될 수 있게 한다.

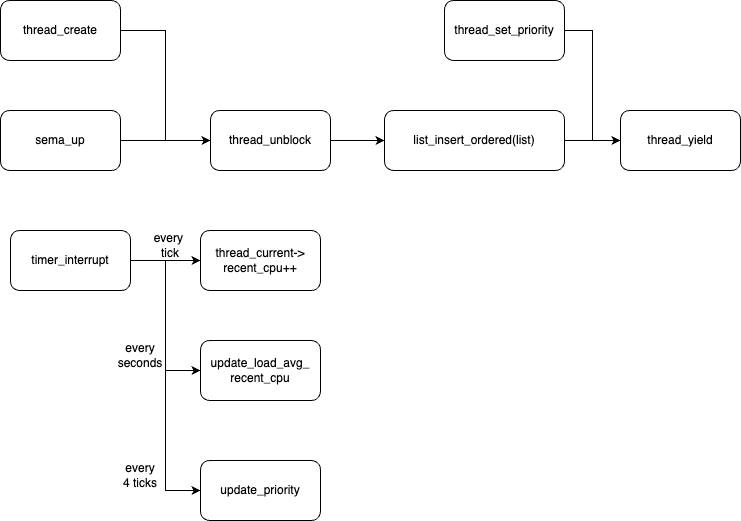
Thread/init.c, thread.h, thread.c에 aging을 위한 새로운 변수를 추가한다. Thread.h 에서 thread의 TCB에 recent\_cpu, nice 값을 생성하고 load\_avg 값을 전역변수로 생성한다. 처음 thread가 생성될 때 값은 0으로 초기화되며 자식 thread가 생성될 때 부모 thread의 recent\_cpu, nice 값은 부모에게 상속받도록 수정한다.

1. **연구 결과**
   1. **Flow Chart**

* II. B. 개발 내용의 각 항목에 대하여 Flow Chart 작성  
  (추가구현에 대해서는 flow chart를 작성하지 않아도 됨)
* Alarm clock



* Priority scheduler

****

* 1. **제작 내용**
* II. B. 개발 내용의 각 항목에 대하여 실질적으로 구현한 코드의 관점에서 작성 (구현 내용, 알고리즘 등을 명확히 서술할 것)
  + 구현에 있어 Pintos에 내장된 라이브러리나 자체 제작한 함수를 사용한 경우 이에 대해서도 설명
* 개발 중 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결한 방식에 대해 설명
* Alarm clock

텍스트이(가) 표시된 사진

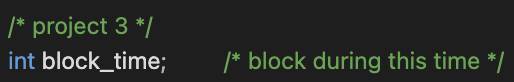
자동 생성된 설명

Devices/timer.h 에서 block list를 관리하기 위해 새로운 list를 추가 선언

텍스트이(가) 표시된 사진

자동 생성된 설명

Devices/timer.c 에서 새로 선언한 list를 list\_init 함수를 이용해 초기화한다.



Thread/thread.h의 TCB에 block\_time이라는 새로운 변수를 추가 선언한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Devices/timer.c의 timer\_sleep 함수에서 기존의 busy wating 방식을 삭제하고 block되기 전에 block\_list에 삽입하고 block될 thread의 block\_time을 저장하고 thread\_block 함수를 이용해 현재 thread를 block state로 변경한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Devices/timer.c 에서 timer\_interrupt 함수를 수정한다. 이 함수는 매 tick마다 interrupt로 실행되며 block\_list를 순회하면서 block time이 다 된(block\_time <= ticks) thread를 thread\_unblock 함수를 이용해 ready state로 바꿔주며 block\_list에서 제거한다.

* Priority Scheduling

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c에 새로운 함수를 추가한다. List\_inset\_ordered 함수에 들어갈 함수 parameter로 사용될 함수이다. 이 함수는 각 thread의 priority를 비교하여 bool 값을 반환한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에서 thread\_yield 함수를 수정한다. 기존에는 CPU 를 양도하고 ready list에 삽입될 때 ready list의 맨 뒤에 삽입되는 Round Robin 방식이었지만 이를 삭제하고 우선순위를 고려하여 삽입하기 위해 list\_insert\_ordered 함수를 이용해 ready list에 삽입한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에서 thread\_unblock 함수를 수정한다. Block state 이던 thread를 ready state로 바꿀 때 ready list에 삽입하는 방식을 thread\_yield와 동일한 방식으로 수정한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c에서 thread\_create 함수를 수정한다. 새로운 thread가 생성되어 thread\_unblock 함수를 통해 ready list에 삽입되어 있는 상태에서 새로 생성된 thread가 현재 running thread보다 높은 priority를 갖고 있는 경우 preemptive 방식이므로 thread\_yield 함수를 호출해 새로 scheduling 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c의 thread\_set\_priority 함수를 수정한다. 현재 running thread의 priority가 기존의 priority보다 낮게 변경될 경우 ready list의 thread들보다 priority가 낮을 가능성이 생기므로 thread\_yield 함수를 이용해 새로 scheduling 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/synch.c의 sema\_up 함수를 수정한다. Sema\_down 함수에서 thread의 priority를 고려하지 않고 waiter list에 삽입했다. Sema\_up 함수에서 기존에는 waiter list의 맨 앞에 있는 thread를 ready state로 만들면서 priority를 고려하지 않았다. 따라서 waiter\_list를 순회하면서 가장 높은 priority를 찾아 해당 thread를 우선적으로 ready state로 만듦으로써 priority를 고려하여 scheduling할 수 있다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/init.c, thread.h, thread.c 에 priority-aging에 필요한 변수들을 선언한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Load\_avg는 thread/thread.c에 전역변수로 선언하고 nice, recent\_cpu는 thread.h의 TCB에 선언한다.

텍스트이(가) 표시된 사진

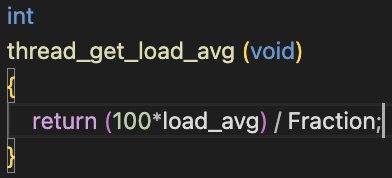
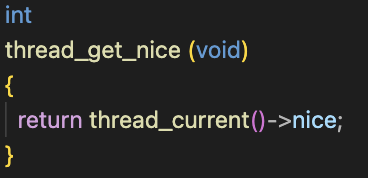
자동 생성된 설명

Thread/thread.c에서 thread\_init 함수를 수정한다. 초기에 thread가 생성될 때 nice, recent\_cpu 값은 0으로 초기화된다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에서 init\_thread 함수를 수정한다. Thread가 생성될 때 부모 thread의 nice, recent\_cpu 값을 상속받는다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에서 thread\_get\_nice|recent\_cpu|load\_avg 함수를 수정한다. 각 value를 반환할 때 recent\_cpu, load\_avg는 100을 곱한 후 fixed point 연산에 맞추어 반환한다.

텍스트이(가) 표시된 사진

자동 생성된 설명텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c에 새로운 함수를 정의한다. Recalculate\_priority는 해당 thread의 nice, recent\_cpu 값을 이용해 priority를 새로 계산하는 함수이다. Recalculate\_recent\_cpu는 load\_avg, nice 값을 이용해 recent\_cpu값을 재계산하는 함수이다. 계산 과정에서 각 변수를 더하고 곱하는 과정에서 fixed-point 연산을 이용한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에 새로운 함수를 정의한다. Update\_load\_avg\_recent\_cpu 함수는 load\_avg, recent\_cpu 값을 update하는 함수이다. List\_size 함수를 이용해 현재 ready list의 크기를 알고 현재 thread가 idle\_thread가 아니라면 값을 1 증가시키고 이 값을 이용해 load\_avg 값을 재계산한다. 기존에 pintos에 정의되어 있던 thread\_foreach 함수를 이용해 모든 thread의 recent\_cpu 값을 재계산한다. Thread\_foreach 함수는 모든 thread를 순회하면서 parameter로 넘어온 함수를 실행시켜주는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Thread/thread.c 에 새로운 함수를 정의한다. Update\_priority 함수는 thread\_foreach 함수를 이용해 모든 thread의 priority를 재계산하는 함수이다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Devices/timer.c에서 timer\_interrupt 함수를 수정한다. Thread\_prior\_aging이 true일 경우(aging이 진행될 경우) 매 tick 마다(timer\_interrupt 함수가 호출될 때마다) 현재 thread의 recent\_cpu 값을 1 증가시키고 매 초마다(100 tick) load\_avg와 모든 thread의 recent\_cpu 값을 재계산한다. 4tick 마다 모든 thread의 priority를 재계산한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

Multi-level Feedback Queue scheduler는 priority를 임의로 변경할 수 없으므로 thread\_set\_priority가 작동하지 못하게 한다.

텍스트이(가) 표시된 사진

자동 생성된 설명

기존에 prior\_aging에서 작동하던 priority, recent\_cpu, load\_avg 계산 과정들이 mlfqs에서도 동일하게 작동할 수 있게 flag를 설정한다.

* 1. **시험 및 평가 내용**
* priority-lifo.c 코드 및 priority-lifo 테스트 결과 분석

텍스트이(가) 표시된 사진

자동 생성된 설명

Priority-lifo는 lifo queue 처럼 schedule될 수 있는 지 test하는 case이다. 동일한 priority를 가진 thread를 16개씩 16번 만드는데 이때 낮은 priority를 가진 thread들부터 먼저 생성되는 반복문이지만 현재 구현한 priority scheduler는 높은 priority를 가진 thread가 우선적으로 schedule되므로 가장 늦게 만들어진 thread가 먼저 simple\_thread\_func를 먼저 실행 완료해서 그 순서를 파악하기 위해 main thread에서 반복문을 돌려서 결과를 확인했다. 결과적으로 LIFO(Last-In, First-out) 형태로 가장 늦게 들어온 데이터를 가장 먼저 처리하는 Test가 잘 진행되었음을 알 수 있다.

* make check 수행 결과를 캡처하여 첨부

텍스트이(가) 표시된 사진

자동 생성된 설명