

1. 프로젝트 개요:

이번 과제에서는 Pintos의 timer_sleep 기능을 개선하여 효율성을 높였습니다. 기존 Pintos에서 설정되어있는 timer_sleep() 함수는 'busy waiting' 방식을 사용하는데, 이는 CPU를 계속 점유하면서 특정 조건이 충족되기 전에는 무한루프를 도는 비효율적인 방식이었습니다. 이러한 방식은 CPU 자원을 비효율적으로 사용하며, 멀티 프로세싱 환경에서는 CPU 사용 시간을 적절히 분배할 수 없다는 단점을 가집니다. 따라서 이를 Blocking방식으로 변경하여 효율성을 높이고자 합니다.

2. 프로젝트 세부 사항 - 알람 클럭 개선:

이러한 문제를 해결하기 위해, timer_sleep() 함수의 구현 방식을 바꾸는 것을 목표로 설정하였습니다. busy waiting 방식 대신, thread가 blocking되어 CPU를 점유하지 않는 'Blocking' 방식을 구현하였습니다. 이를 위해 thread.h의 구조체를 수정하였고, thread.c 및 timer.c에서 관련 기능을 개선하였습니다.

3. 수정된 데이터 구조:

thread.h에서 스레드 구조체에 wakeup_time 변수를 추가하여 스레드가 깨어날 시간을 저장하도록 했습니다. 이 변수는 timer_sleep() 함수에서 설정되며, timer_interrupt() 함수에서 확인할 수 있습니다.

4. 수정된 코드 설명

4-1) 새로운 변수를 thread.h에 선언하였습니다.

```
struct thread
{
    /* Owned by thread.c. */
    tid_t tid; /* Thread identifier. */
    enum thread_status status; /* Thread state. */
    char name[16]; /* Name (for debugging purposes). */
    uint8_t *stack; /* Saved stack pointer. */
    int priority; /* Priority. */
    struct list_elem allelem; /* List element for all threads list. */

    /* Shared between thread.c and synch.c. */
    struct list_elem elem; /* List element. */

    /* code change part */
    int64_t wakeup_time;
}
```

4-2) 'timer.c'에 있는 **timer_sleep** 함수를 수정하였습니다. 스레드를 블록하고 지정된 시간이 지난 후 thread가 깨어나도록 시간을 설정해 주었습니다.

```
void
timer_sleep (int64_t ticks)
{
    int64_t start = timer_ticks ();
    struct thread *cur = thread_current(); //sj

    enum intr_level old_level;
    old_level = intr_disable();

    cur->wakeup_time = start + ticks;
    list_insert_ordered(&timer_wait_list, &cur->elem, less_wakeup_time, NULL);
    thread_block();

    intr_set_level(old_level);
}
```

설명:

위 코드에서는 두 가지 주요 변경 사항이 있습니다.

4-2-1) **wakeup_time** 변수를 사용하여 스레드가 깨어날 시간을 카운트합니다.(이 변수는 스레드 구조체에 추가되어 있음)

4-2-2) **thread_block()** 함수를 호출하여 현재 스레드를 block합니다. CPU를 불필요하게 소비하는 busy-waiting 방식을 개선하여 다른 스레드를 실행하게 합니다.

4-3) 'timer.c'에 있는 **timer_interrupt** 함수를 수정하였습니다. 각 인터럽트 때마다 timer_wait_list 에 있는 스레드들의 wakeup_time을 확인하고, 깨어날 시간이 되면 해당 thread를 깨우는, unblock 하는 코드입니다.

```

/* Timer interrupt handler. */
static void
timer_interrupt (struct intr_frame *args UNUSED)
{
    ticks++;
    thread_tick ();

    enum intr_level old_level;
    old_level = intr_disable();

    while(!list_empty (&timer_wait_list)) {
        struct list_elem *e = list_front(&timer_wait_list); //sj
        struct thread *t = list_entry(e, struct thread, elem);

        if(t -> wakeup_time > ticks)
            break

        list_remove(e); //sj
        thread_unblock(t); //sj
    }

    intr_set_level(old_level);
}

```

설명: 차단된 스레드 목록(blocked_list)을 반복하면서 각 스레드가 깨어나야 하는 시간(wakeup_ticks)이 현재 시간(ticks)과 동일하거나 그 이상인 경우, 해당 스레드를 차단 해제하고 목록에서 제거합니다.

매 타이머 인터럽트마다 ticks 값을 증가시키고, 각 스레드가 sleep 중이면서 wakeup_ticks 값을 초과하거나 동일한 경우에 대해 차단 해제하고 차단 목록에서 제거합니다.

4-4) 'timer.c'에 static bool less_wakeup_time 함수를 추가했습니다.(스레드의 깨어나야 할 시간(wakeup_time)이 a 스레드보다 b 스레드가 더 작은지 확인하는 함수) 만약 a 스레드의 깨어나야 할 시간이 b 스레드보다 작다면 true를 반환하고, 그렇지 않다면 false를 반환합니다.

```

static bool
less_wakeup_time(const struct list_elem *a_, const struct list_elem *b_, void *aux UNUSED)
{
    const struct thread *a = list_entry(a_, struct thread, elem);
    const struct thread *b = list_entry(b_, struct thread, elem);

    return a->wakeup_time < b->wakeup_time;
} //sj

```

설명: wakeup_time이 더 적은 스레드를 찾는 데 사용. a스레드와 b스레드의 wakeup_time 을 비교하여 a스레드의 wakeup_time 이 더 적다면 true 를 반환하고, 그렇지 않으면 false 를 반환합니다. 스레드 목록을 wakeup_time 기준으로 정렬할 때 필요한 함수입니다.

5. 결과

```
pintos@pintos-VirtualBox:~/pintos/src/threads$ make check
cd build && make check
make[1]: Entering directory '/home/pintos/pintos/src/threads/build'
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
FAIL tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
FAIL tests/threads/priority-change
FAIL tests/threads/priority-donate-one
FAIL tests/threads/priority-donate-multiple
FAIL tests/threads/priority-donate-multiple2
FAIL tests/threads/priority-donate-nest
FAIL tests/threads/priority-donate-sema
FAIL tests/threads/priority-donate-lower
FAIL tests/threads/priority-fifo
FAIL tests/threads/priority-preempt
FAIL tests/threads/priority-sema
FAIL tests/threads/priority-condvar
FAIL tests/threads/priority-donate-chain
FAIL tests/threads/mlfqs-load-1
FAIL tests/threads/mlfqs-load-60
FAIL tests/threads/mlfqs-load-avg
FAIL tests/threads/mlfqs-recent-1
pass tests/threads/mlfqs-fair-2
pass tests/threads/mlfqs-fair-20
FAIL tests/threads/mlfqs-nice-2
FAIL tests/threads/mlfqs-nice-10
FAIL tests/threads/mlfqs-block
20 of 27 tests failed.
../../tests/Make.tests:26: recipe for target 'check' failed
```

5-2) squish-pty후 Test 실행

```
pintos@pintos-virtlab[Box]:~/pintos/src$ head -n 910 pintos -q run alarm-multiple
Use of literal control characters in variable names is deprecated at /home/pintos/pintos/src/utils/pintos line 911.
Prototype mismatch: sub main: SIGVTALRM () vs none at /home/pintos/pintos/src/utils/pintos line 935.
Constant subroutine SIGVTALRM redefined at /home/pintos/pintos/src/utils/pintos line 927.
aquilah-ply bochs -q
-----
        Bochs x86 Emulator 2.6.2
    Built from SVN snapshot on May 26, 2013
    Compiled on Feb 19 2819 at 17:39:30
-----
0000000000000000|      | reading configuration from bochsrc.txt
0000000000000000|      | bochsrc.txt:8: 'user.shortcut' will be replaced by new 'keyboard' option.
0000000000000000|      | installing qemu module as the Bochs GUI
0000000000000000|      | using log file bochsout.txt
Milo hdat
Loading.....
Kernel command line: q run alarm-multiple
Pintox booting with 4,096 KB RAM...
383 pages available in kernel pool.
383 pages available in user pool.
Calibrating timer.... 201,600 loops/s.
Boot complete.
```

```

Boot complete.
Executing 'alarm-multiple':
(alarm-multiple) begin
(alarm-multiple) Creating 5 threads to sleep 7 times each.
(alarm-multiple) Thread 0 sleeps 18 ticks each time.
(alarm-multiple) Thread 1 sleeps 28 ticks each time, and so on.
(alarm-multiple) If successful, product of iteration count and
(alarm-multiple) sleep duration will appear in nondescending order.
(alarm-multiple) thread 0: duration=18, iteration=1, product=18
(alarm-multiple) thread 1: duration=28, iteration=1, product=28
(alarm-multiple) thread 0: duration=18, iteration=2, product=36
(alarm-multiple) thread 2: duration=38, iteration=1, product=38
(alarm-multiple) thread 0: duration=18, iteration=3, product=54
(alarm-multiple) thread 3: duration=48, iteration=1, product=48
(alarm-multiple) thread 1: duration=28, iteration=2, product=56
(alarm-multiple) thread 0: duration=18, iteration=4, product=72
(alarm-multiple) thread 4: duration=58, iteration=1, product=58
(alarm-multiple) thread 0: duration=18, iteration=5, product=90
(alarm-multiple) thread 2: duration=38, iteration=2, product=76
(alarm-multiple) thread 1: duration=28, iteration=3, product=84
(alarm-multiple) thread 0: duration=18, iteration=6, product=108
(alarm-multiple) thread 0: duration=18, iteration=7, product=126
(alarm-multiple) thread 3: duration=48, iteration=2, product=96
(alarm-multiple) thread 1: duration=28, iteration=4, product=112
(alarm-multiple) thread 2: duration=38, iteration=3, product=114
(alarm-multiple) thread 4: duration=58, iteration=2, product=116
(alarm-multiple) thread 1: duration=28, iteration=5, product=140
(alarm-multiple) thread 3: duration=48, iteration=3, product=120
(alarm-multiple) thread 2: duration=38, iteration=4, product=126
(alarm-multiple) thread 1: duration=28, iteration=6, product=126
(alarm-multiple) thread 4: duration=58, iteration=3, product=130
(alarm-multiple) thread 2: duration=38, iteration=5, product=130
(alarm-multiple) thread 3: duration=48, iteration=4, product=168
(alarm-multiple) thread 2: duration=38, iteration=6, product=186
(alarm-multiple) thread 4: duration=58, iteration=4, product=200
(alarm-multiple) thread 3: duration=48, iteration=5, product=200
(alarm-multiple) thread 2: duration=38, iteration=7, product=210
(alarm-multiple) thread 3: duration=48, iteration=6, product=240
(alarm-multiple) thread 4: duration=58, iteration=5, product=240
(alarm-multiple) thread 3: duration=48, iteration=7, product=280
(alarm-multiple) thread 4: duration=58, iteration=6, product=300
(alarm-multiple) thread 4: duration=58, iteration=7, product=330
(alarm-multiple) end
Execution of 'alarm-multiple' complete.
User: 850 ticks
Thread: 558 idle ticks, 382 kernel ticks, 0 user ticks
Console: 2952 characters output
Keyboard: 8 keys pressed

```

```

(alarm-multiple) end
Execution of 'alarm-multiple' complete.
User: 850 ticks
Thread: 558 idle ticks, 382 kernel ticks, 0 user ticks
Console: 2952 characters output
Keyboard: 0 keys pressed
Powering off....

```

```

=====
Bochs is exiting with the following message:
[UNIMP ] Shutdown port: shutdown requested
=====

```

성공적으로 동작했음을 알 수 있습니다.