Operating System

project 4

2009004076 유상현
2013011446 손윤하

4-1 Alarm System Call

1. 과제 개요

busy waiting으로 구현되어있는 pintos sleep 기능을 timer interrupt 기반으로 수정하여 성능을 높힌다.

2. 과제 목표

- sleep 함수 내에서 while 문을 사용하지 않고, sleeplist 를 사용하여 block 시킨다.
- 깨어날 때는 timer interrupt를 이용하고 여러개의 타이머를 관리하는 overhead를 줄인다.
- 타이머의 갯수에 비례하여 낭비하는 자원을 증가시키지 않는다.

3. 과제 해결을 위한 Pintos 분석 내용

현 상태에서 sleep함수는 while문을 이용하여 구현되어 있다. 계속 이 상태에서 blocking되어 있고 매 실행마다 현재 time tick을 비교해야 하기 때문에 성능이 낭비된다. 또한 여러개의 타이머가 동작한다면 필요한 자원이 선형적으로 증가한다.

pintos OS는 주기적으로 timer interrupt를 발생시키기 때문에 이를 이용하여 timer를 체크할 수 있다.

4. 해결 과정

1. sleep을 호출한 thread를 block state로 전이시킨 sleep_list에 insert한다. 이러한 상태에선 ready_list에 진입하지 않는다. 따라서 thead는 실행되지 않는다.

2. sleep_list는 현재 sleep 중인 모드 쓰레드들이 보관, 관리되는데 이 리스트는 tick에 따라 오름차순으로 미리 정렬하여 매틱 체크할 때 성능저하를 막는다. (가장 작은 tick값을 caching한다.)

3. timer interrupt가 주기적으로 호출되는데, 이 때 caching한 tick과 비교하여 awake시켜야 할 thread가 있다면 해당 thread를 sleep_list에서 제거하고 ready state로 전이시킨 후 ready_list에 insert한다.

4-2 Priority Scheduling

1. 과제 개요

priority를 고려하지 않고 모두 공평하게 동작하고 있는 현재의 round-robin scheduler를 수정하여 priority가 높은 쓰레드를 우선하여 실행하도록 한다.

2. 과제 목표

- ready_list에 thread를 insert할 때 priority를 고려하여 정렬-삽입 하여 가장 priority가 높은 thread부터 순차적으로 실행되도록 한다.

3. 과제 해결을 위한 Pintos 분석 내용

- Pintos OS는 현재 관련 기능이 구현되지 않았지만 thread 를 생성할 때 priority를 인자로 전달 받고, thread struct 내에 보관하고 있다.
- ready_list는 단순(비정렬) list로써 정렬되어있지 않다.

- 모든 thread는 일정 주기 마다 공평하게 scheduling되고 있다.(효율성은 고려하지 않음)
- thread의 priority는 생성 후에도 변경될 수 있다.
- lock과 관련해서 역시 priority가 고려되어 있지 않다.

4. 해결 과정

1. ready_list에 insert 할 때 해당 thread의 priority에 따라 정렬하여 삽입한다. ready_list로부터 다음 실행될 thread가 선택될 때 priority가 가장 높은 thread가 선택되도록 한다.

2. ready_list에서 scheduling될 때 뿐만 아니라 생성시, priority 변경시에도 우선순위를 고려하여 현재 실행 중인 thread보다 우선순위가 높다면 실행을 선점하도록 한다.

4-3 Priority Scheduling and Synchronization

1. 과제 개요

lock, semaphore, condition variable등 thread동기화와 관련된 객체들에서 thread를 관리할 때 priority를 고려하여 정렬, 사용하도록 한다.

2. 과제 목표

- semaphore를 대기하고 있는 thread(waiter) 들은 시도한 순서대로 acquire하도록 관리되고 있다. 이를 4-2의 내용과 유사하게 해당 thread의 priority를 내림차순으로 정렬하여 순서대로 깨어날 수 있도록 한다.

- 마찬가지로 condition variable의 waiter들 역시 priority를 고려하여 정렬하여 순서대로 깨어날 수 있도록 한다.

3. 과제 해결을 위한 Pintos 분석 내용

- thread 동기화를 위해 semapohre 및 condition variable을 사용하고 있다.
- 현재 semaphore 및 condition variable의 list는 비정렬 상태로 관리되고 있거나 구현되어 있지 않다.

4. 해결 과정

1. sema_down시 waiters list에 thread의 priority를 고려하여 삽입한다.
2. sema_up 시에는 이미 정렬되어있으므로 waiters로부터 front를 pop하여 해당 thread를 우선적으로 실행한다.
3. condition variable을 기다릴 때도 cond_wait함수에서 semaphore 들도 그 내부의 waiters 리스트의 thread의 front의 priority를 고려하여 정렬 해서 삽입한다.
4. cond_signal 함수에서 waiter를 깨울 떄에도 가장 앞의 waiter를 pop하여 실행한다.

4-4 Priority inversion problem

1. 개요

높은 우선순위 쓰레드가 낮은 우선순위 쓰레드에 Lock를 요청했으나, 중간 우선순위 쓰레드가 도중에 Lock을 요청하면 우선순위의 역전이 발생할 수 있다.

2. 목표

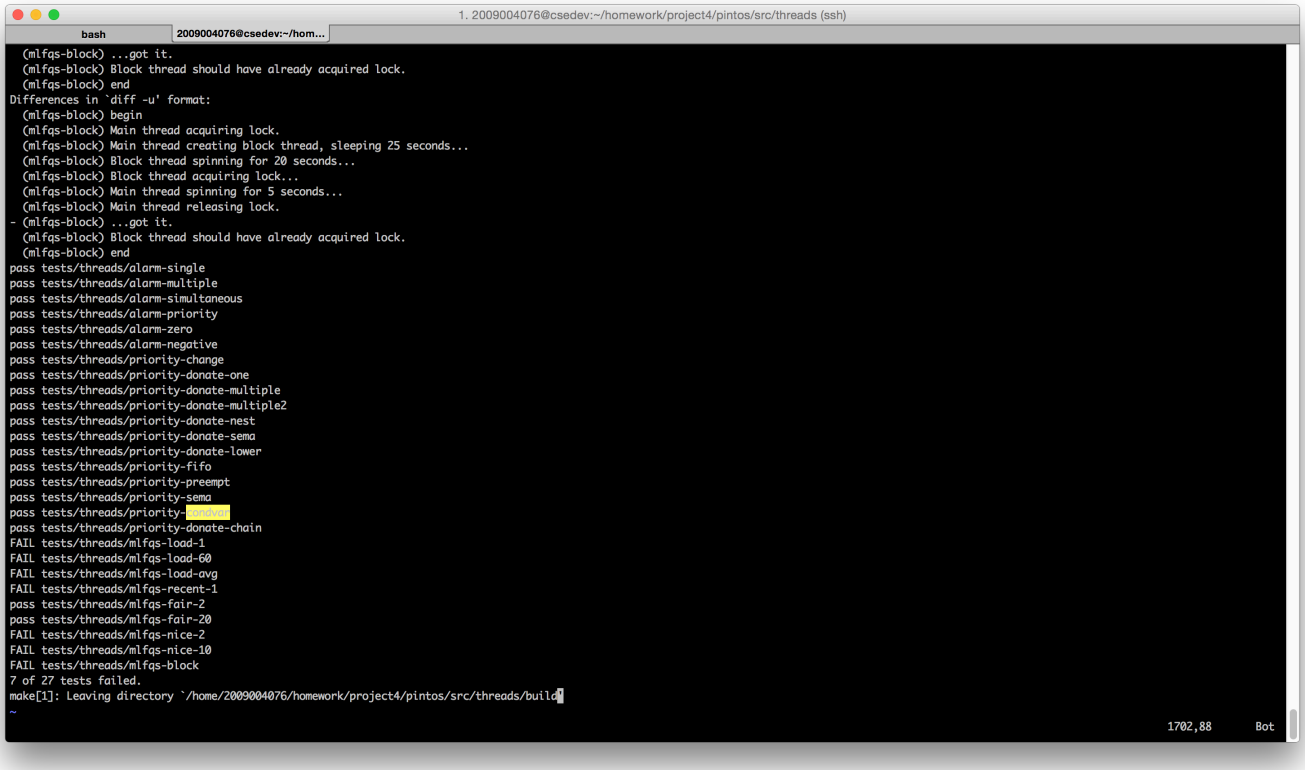
- Priority donation 구현: Lock을 요청함과 동시에 낮은 우선순위의 쓰레드에 자신의 우선순위를 기부(donate)하도록 한다.
- Multiple donation 구현: 쓰레드 L이 Lock A, Lock B를 갖고 있고 쓰레드 M이 Lock A를, 쓰레드 H가 Lock B를 요청하면 총 두 차례의 우선순위 기부가 발생한다(우선순위 L -> M -> H). 이때 Lock B가 해제되면 쓰레드 L의 우선순위가 초기값 L이 아닌 쓰레드 M의 우선순위 M으로 돌아되도록 수정해야 한다.
- Nested donation 구현: 쓰레드 H가 쓰레드 M의 Lock을 요청하고, 쓰레드 M이 쓰레드 L의 Lock을 요청하면 쓰레드 H의 우선순위가 쓰레드 M, L에 모두 기부되도록 한다.

3. 과제 해결을 위한 Pintos 분석

- 현재는 lock을 요청받으면 기부 과정 없이 그대로 sema_down을 호출하여 락을 거는데, 이 과정에서 기부를 해야한다.
- 이미 lock의 holder가 존재할 경우 도네이션 리스트에 쓰레드를 삽입하고 기부를 수행하도록 변경해야 한다.

4. 해결과정

1. struct thread에 관련 항목들을 추가하고 초기화 코드를 작성한다.
2. 기부 과정에 필요한 함수들을 작성한다.
3. 기존 lock 관련 함수들에서 기부 관련 함수를 호출해주고, multiple donation과 nested donation 등으로 꾸준히 우선순위가 변경되므로 적절한 순서로 스케쥴링 처리를 해준다.


1. 2009004076@csedev:~/homework/project4/pintos/src/threads (ssh)
bash
2009004076@csedev:~/hom...
(mlfqs-block) ...got it.
(mlfqs-block) Block thread should have already acquired lock.
(mlfqs-block) end
Differences in `diff -u' format:
(mlfqs-block) begin
(mlfqs-block) Main thread acquiring lock.
(mlfqs-block) Main thread creating block thread, sleeping 25 seconds...
(mlfqs-block) Block thread spinning for 20 seconds...
(mlfqs-block) Block thread acquiring lock...
(mlfqs-block) Main thread spinning for 5 seconds...
(mlfqs-block) Main thread releasing lock.
- (mlfqs-block) ...got it.
(mlfqs-block) Block thread should have already acquired lock.
(mlfqs-block) end
pass tests/threads/alarm-single
pass tests/threads/alarm-multiple
pass tests/threads/alarm-simultaneous
pass tests/threads/alarm-priority
pass tests/threads/alarm-zero
pass tests/threads/alarm-negative
pass tests/threads/priority-change
pass tests/threads/priority-donate-one
pass tests/threads/priority-donate-multiple
pass tests/threads/priority-donate-multiple2
pass tests/threads/priority-donate-nest
pass tests/threads/priority-donate-sema
pass tests/threads/priority-donate-lower
pass tests/threads/priority-fifo
pass tests/threads/priority-preempt
pass tests/threads/priority-sema
pass tests/threads/priority-condvar
pass 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
7 of 27 tests failed.
make[1]: Leaving directory `/home/2009004076/homework/project4/pintos/src/threads/build'
~
1702,88
Bot