Project 02

Implementing simple system schedulers on xv6

Due date:
2024. 04. 21. 11:59pm

xv6 Scheduler

- 스케줄링은 다중 프로그램 운영체제의 기본이 되며,
  실행 가능한 프로세스 중 가장 적합한 프로세스를 선택하여 실행합니다.
  이를 통해 운영체제가 주어진 자원을 더 효율적으로 사용할 수 있도록 합니다.

- xv6에서는 기본적으로 매우 간단한 버전의 Round-Robin 방식의 스케줄러를 사용합니다.

- 이번 과제에서는 이론 시간에 배운 스케줄러들을 xv6에 구현해보는 것을 목적으로 하며,
  이러한 과정에서 실제 동작하는 스케줄러들을 디자인하고 구현, 테스트, 분석해봅시다.

Default RR Scheduler

- xv6의 스케줄러는 기본적으로 Round-Robin 정책을 사용하고 있으며,
  기본 동작은 다음과 같습니다.

  - 타이머 인터럽트가 발생하면,
    현재 실행중인 프로세스를 RUNNABLE 상태로 전환시키고,
    프로세스의 배열에서 다음 인덱스의 프로세스를 실행시킵니다.
    배열의 마지막 프로세스까지 실행시켰다면 배열의 처음 프로세스부터 다시 시작합니다.

  - 타이머 인터럽트가 발생하는 주기를 tick이라 하며, 기본 값은 약 10ms 입니다.
    즉, 약 10ms마다 한 번씩 프로세스 간의 context switch가 발생합니다.

Scheduler

- xv6에 구현할 사항들은 다음과 같습니다.

1. MLFQ(Multi-Level Feedback Queue) part
   - Round-Robin scheduling
   - Priority scheduling
2. MoQ(Monopoly Queue) part
   - FCFS scheduling

Specification - MLFQ

- **4-level feedback queue**
- L0~L3, 총 4개의 큐로 이루어져 있고, 수가 작을수록 우선순위가 높은 큐입니다.
- L_i 큐는 2i + 2 ticks의 time quantum을 가집니다.
- 처음 실행된 프로세스는 모두 가장 높은 레벨의 큐(**L0**)에 들어갑니다.

Specification - MLFQ

- **L0** 큐, **L1** 큐 그리고 **L2** 큐는 기본 RR정책을 따릅니다.
  - 스케줄러는 기본적으로 L0 큐의 RUNNABLE한 process를 스케줄링 합니다.
  - L0 큐의 RUNNABLE한 프로세스가 없을 경우, L1의 process를 스케줄링 합니다.
  - L1 큐의 RUNNABLE한 프로세스가 없을 경우, L2의 process를 스케줄링 합니다.

- L0 큐에서 실행된 프로세스가 time quantum을 모두 사용했을 때,
  - pid가 홀수인 프로세스들은 **L1** 큐로 내려가고, time quantum을 초기화합니다.
  - pid가 짝수인 프로세스들은 **L2** 큐로 내려가고, time quantum을 초기화합니다.
- L1 및 L2 큐에서 실행된 프로세스가 time quantum을 모두 사용했을 때,
  프로세스는 L3 큐로 내려가고, time quantum을 초기합니다.

Specification - MLFQ

- **L3** 큐는 priority 스케줄링을 합니다.
  - L1 및 L2 큐의 RUNNABLE한 프로세스가 없을 경우,
    스케줄러는 L3의 process를 스케줄링 합니다.
  - 같은 큐 내에서는 priority가 높은 프로세스가 먼저 스케줄링 되어야 합니다.
  - priority를 변경할 수 있는 시스템 콜인 setpriority가 추가되어야 합니다.
  - Priority는 프로세스마다 별개로 존재하며, 0 이상 10 이하의 정수 값을 가집니다.
    - Priority 값이 클수록 우선순위가 높습니다.
    - 처음 프로세스가 실행될 때 priority는 0으로 설정합니다.
    - Priority가 같은 프로세스끼리는 어느 것을 먼저 스케줄링해도 괜찮습니다.
  - Priority 값은 L3 큐에서만 유효하며, L0~L2 큐에서는 아무런 역할이 없습니다.

Specification - MLFQ

- **L3** 큐에서 실행된 프로세스가 time quantum을 모두 사용한 경우, 해당 프로세스의 priority가 1 감소하며, time quantum은 초기화됩니다. (단, 0에서는 더 감소하지 않습니다.)

- Starvation을 막기 위해 priority boosting을 구현해야 합니다.

- Priority boosting
  - Global tick이 100 ticks가 될 때마다 모든 프로세스들은 L0 큐로 재조정됩니다.
  - Priority boosting이 될 때, 모든 프로세스들의 time quantum은 초기화됩니다.

Specification - MLFQ

- 타이머 인터럽트나 yield, sleep 등을 통해 스케줄러에게 실행 흐름이 넘어올 때마다,
  스케줄러는 RUNNABLE한 프로세스가 존재하는 큐 중 가장 레벨이 높은 큐를 선택해야 합니다.

Specification – MoQ

- 사용자의 요청에 따라 CPU 자원을 길게 사용해야 할 프로세스가 있을 수 있습니다.
- 해당 프로세스들의 스케줄링을 위해 FCFS 정책을 따르는 Monopoly-큐(MoQ)가 존재합니다.
- MoQ에 속한 프로세스들은 평소에는 스케줄링 되지 않다가 monopolize 시스템콜이 호출되면 즉시 스케줄링 됩니다.
  - 먼저 큐에 들어온 프로세스가 먼저 스케줄링 되어야 합니다.
  - MoQ 큐를 스케줄링 하는 동안 priority boosting은 발생하지 않습니다.
- 프로세스는 setmonopoly 시스템 콜을 통해 MoQ로 이동합니다.
- MoQ의 모든 프로세스들이 종료되면 자동으로 unmonopolize 시스템콜이 호출되며, 기존의 MLFQ part로 돌아갑니다.
- unmonopolize 시스템 콜이 호출되면, Global tick은 0으로 초기화됩니다.

Specification – common

- 모든 preemption은 10ms(1tick) 안으로 일어나면 됩니다.

- Coding Convention은 xv6의 기본적인 형태를 따릅니다.
  - 들여쓰기, 중괄호 위치 등

- 구현의 편의성을 위해 cpu의 개수는 하나임을 가정합니다.
  - 테스트를 할 때는 make의 인자로 CPUS=1을 주거나,
    qemu의 옵션으로 –smp=1을 주고 테스트하시면 됩니다.

- 유저프로세스가 잘못된 행동을 하여 강제 종료를 시키는 경우,
  강제로 종료 시켰다는 메시지를 출력해야 하며, 그 외의 문제가 발생해서는 안됩니다.
  또한 종료 시킨 이후에는 다음 프로세스가 정상적으로 스케줄링 되어야 합니다.

Required system calls

- 다음의 시스템 콜들은 주어진 명세대로 구현해야 합니다.

- **void yield(void)**
  - 자신이 점유한 cpu를 양보합니다.

- **int getlev(void)**
  - 프로세스가 속한 큐의 레벨을 반환합니다.
  - MoQ에 속한 프로세스인 경우 99를 반환합니다.

- **int setpriority(int pid, int priority)**
  - 특정 pid를 가지는 프로세스의 priority를 설정합니다.
  - priority 설정에 성공한 경우 0을 반환합니다.
  - 주어진 pid를 가진 프로세스가 존재하지 않는 경우 -1을 반환합니다.
  - priority가 0 이상 10 이하의 정수가 아닌 경우 -2를 반환합니다.

Required system calls

- **int setmonopoly(int pid, int password)**
  - 특정 pid를 가진 프로세스를 MoQ로 이동합니다. 인자로 독점 자격을 증명할 암호(자신의 학번)을 받습니다.
  - 암호가 일치할 경우, MoQ의 크기를 반환합니다.
  - 존재하지 않는 프로세스의 pid인 경우 -1을 반환합니다.
  - 암호가 일치하지 않는 경우 -2를 반환합니다.
  - 이미 MoQ에 존재하는 프로세스인 경우 -3을 반환합니다.
  - 자기 자신을 MoQ로 이동시키려 하는 경우 -4를 반환합니다.
- **void monopolize()**
  - MoQ의 프로세스가 CPU를 독점하여 사용하도록 설정합니다.
- **void unmonopolize()**
  - 독점적 스케줄링을 중지하고 기존의 MLFQ part로 돌아갑니다.

Evaluation

- **Completeness** : 명세의 요구조건에 맞게 xv6가 올바르게 동작해야 합니다.
- **Defensiveness** : 발생할 수 있는 예외 상황에 대처할 수 있어야 합니다.
- **Wiki & Comment** : 테스트 프로그램과 위키를 기준으로 채점이 진행되므로,
  위키는 최대한 상세히 작성되어야 합니다.
- **Deadline** : 마감 기한을 반드시 지켜야 하며, 마감 기한 이전에 마지막으로 제출된 코드를 기준으로 채점됩니다.
- **DO NOT SHARE AND COPY !!**

Wiki

- **Design** : 명세에서 요구하는 조건에 대해서 **어떻게 구현할 계획**인지, 어떤 자료구조와 알고리즘이 필요한지, **자신만의 디자인**을 서술합니다.
- **Implement** : 실제 구현 과정에서 **변경하게 되는 코드영역**이나 **작성한 자료구조** 등에 대한 설명을 구체적으로 서술합니다.
- **Result** : 컴파일 및 **실행 과정**과, 해당 명세에서 요구한 부분이 정상적으로 동작하는 **실행 결과**를 첨부하고, 이에 대한 동작 과정에 대해 설명합니다.
- **Trouble shooting** : 과제 수행 도중 **발생했던 문제**와 이에 대한 **해결 과정**을 서술합니다. 문제를 해결하지 못했다면 **어떻게 해결하려 하였는지**에 대해 서술합니다.
- **and whatever you want :)**

Submission

- **Design** : 명세에서 요구하는 조건에 대해서 **어떻게 구현할 계획**인지, 어떤 자료구조와 알고리즘이 필요한지, **자신만의 디자인**을 서술합니다.
- **Implement** : 실제 구현 과정에서 **변경하게 되는 코드영역**이나 **작성한 자료구조** 등에 대한 설명을 구체적으로 서술합니다.
- **Result** : 컴파일 및 **실행 과정**과, 해당 명세에서 요구한 부분이 정상적으로 동작하는 **실행 결과**를 첨부하고, 이에 대한 동작 과정에 대해 설명합니다.
- **Trouble shooting** : 과제 수행 도중 **발생했던 문제**와 이에 대한 **해결 과정**을 서술합니다. 문제를 해결하지 못했다면 **어떻게 해결하려 하였는지**에 대해 서술합니다.
- **and whatever you want :)**

Submission

- 구현한 코드와 wiki를 LMS 과제란에 제출합니다.
- WiKi 파일과 xv6-public 파일 전체를 한 디렉토리에 포함시킨 후, 압축하여 제출합니다.
- 제출할 디렉토리의 이름은 "**OS_project02_[수업번호]_[이름]_[학번]**" 입니다.
- Wiki 파일의 이름은 "**OS_project02_[수업번호]_[이름]_[학번].pdf**" 입니다.
  - 제출할 디렉토리의 양식 예시는 다음과 같습니다.
    - ex) OS_project02_12345_주정현_2022123123

      -- xv6-public

      -- OS_project02_12345_주정현_2022123123.pdf
    - 해당 디렉토리를 압축하여 OS_project01_12345_주정현_2022123123.zip 파일을 제출합니다.
- **제출 양식을 엄수하고 의미 없는 파일 및 디렉토리는 반드시 제거하여 제출하시길 바랍니다.**
- **제출 기한: 2024년 4월 21일 11:59 pm (No additional submission)**

Thank you