Operating System Project #1 - xv6 MLFQ scheduler report

201811061 박동진

-Description of implementation

1. proc.h의 struct proc에 process의 남은 tick을 저장하는 int tick_remain과 process가 어느 큐(top queue, middle queue, bottom queue)에 있는지 저장하는 int level을 추가하였습니다.
2. proc.c에 global하게 사용할 수 있는 2차원 queue 배열 queue[3][NPROC]을 선언하였습니다. queue[0][NPROC]이 top queue, queue[1][NPROC]이 middle queue, queue[2][NPROC]이 bottom queue로 사용되었습니다.
3. 각 queue의 enqueue index를 global하게 선언하여 enqueue, dequeue 함수를 편리하게 사용할 수 있도록 하였습니다. 각 queue는 처음에는 비어 있기 때문에 0으로 초기화 하였습니다. (top_idx, mid_idx, bot_idx)
4. enqueue(int level, struct proc* p) 함수를 통해 원하는 queue level에 proc* p 를 삽입할 수 있도록 하였습니다.
5. dequeue(int level) 함수를 통해 원하는 queue level에서 dequeue를 할 수 있도록 하였습니다.
6. proc.c의 process가 새로 초기화되는 allocproc() 함수에 top queue에 process를 집어넣기 위해 tick_remain 을 1 level을 0으로 초기화 하고 top queue에 enqueue 하였습니다.
7. schduler() 함수에서 ptable 대신 queue를 scan 하였습니다. process가 선택될 때마다 tick_remain을 하나씩 줄임으로써 process의 level이 변경되어야 하는지, 한번 더 실행되어야 하는지를 결정하였습니다.
8. process가 다시 scheduler로 돌아 왔을 때, process가 runnable하지 않은 경우, queue level에 따라 tick_remain을 재설정함으로써, scheduler에 의해 process가 다시 선택되었을 때, queue에서 허용한 tick 수만큼 실행이 가능하도록 하였습니다.
9. process가 middle queue 혹은 bottom queue에서 지속적으로 실행될 때, 매 tick마다 상위 queue에 process가 존재 하는지 확인하여 상위 queue에 process가 있다면 상위 queue로 이동하도록 하였습니다.

-Analysis of benchmark programs including comparison with xv6 default scheduler

1. bench1

* default

```
$ bench1
# of task: 10, Completion time of each task: 0 (tick)
Time from fork() to wait(): 2
(Child) Total response time: 9 (avg: 0), Total turnaround time: 9 (avg: 0)
```

* MLFQ

```
$ bench1
# of task: 10, Completion time of each task: 0 (tick)
Time from fork() to wait(): 3
(Child) Total response time: 9 (avg: 0), Total turnaround time: 9 (avg: 0)
```

* analysis : 1 tick이 채 걸리지 않는 task를 실행했을 때, MLFQ는 top queue에서만 동작하므로 default scheduler와 performance의 차이가 없습니다.

2. bench2

* default

```
$ bench2
# of task: 10, Completion time of each task: 4 (tick)
Time from fork() to wait(): 41
(Child) Total response time: 58 (avg: 5), Total turnaround time: 365 (avg: 36)
```

* MLFQ

```
$ bench2
# of task: 10, Completion time of each task: 4 (tick)
Time from fork() to wait(): 41
(Child) Total response time: 34 (avg: 3), Total turnaround time: 338 (avg: 33)
```

* analysis : MLFQ에서 모든 process가 middle queue에서 종료됩니다. 4 tick task가 middle queue에서 연속적으로 실행되므로 response time과 turnarount time이 향상되었습니다.

3. bench3

* default

```
$ bench3
# of task: 10, Completion time of each task: 16 (tick)
Time from fork() to wait(): 161
(Child) Total response time: 60 (avg: 6), Total turnaround time: 1573 (avg: 157)
```

* MLFQ

```
$ bench3
# of task: 10, Completion time of each task: 16 (tick)
Time from fork() to wait(): 161
(Child) Total response time: 36 (avg: 3), Total turnaround time: 1260 (avg: 126)
```

* analysis : MLFQ에서 16 tick task가 bottom queue에서 연속적으로 실행되므로 response time과 turnaround time이 향상되었습니다.

4. bench4

* default

```
$ bench4
# of task: 10, Completion time of each task: [0,5,10,15,20,0,..] (tick)
Time from fork() to wait(): 101
(Child) Total response time: 36 (avg: 3), Total turnaround time: 600 (avg: 60)
```

* MLFQ

```
$ bench4
# of task: 10, Completion time of each task: [0,5,10,15,20,0,..] (tick)
Time from fork() to wait(): 102
(Child) Total response time: 36 (avg: 3), Total turnaround time: 549 (avg: 54)
```

* analysis : workload가 큰 task를 연속적으로 실행하므로 turnaround time이 향상된 것을 알 수 있습니다.

5. bench5

* default

```
$ bench5
# of task: 50, Completion time of each task: one [200] and other [0,3,6,9,12,0,..]
turnaround time of long task: 506
Time from fork() to wait(): 506
```

* MLFQ

```
$ bench5
# of task: 50, Completion time of each task: one [200] and other [0,3,6,9,12,0,..]
turnaround time of long task: 505
Time from fork() to wait(): 505
```

* analysis : workload가 큰 task는 bottom queue로 내려가기 때문에 다른 task가 계속 들어온다면 starvation이 발생합니다. 따라서 turnaround time of long task는 향상 되지 않았습니다.