운영체제 Project 1

202111057 문서흔

1. 변수 추가

```
#define LITTLE_CORE 0 // (added)
#define BIG_CORE 1 // (added)
```

Proc.h에 먼저 Little core를 0으로, Big core를 1로 정의.

```
int cpu_affinity; // 프로
uint runtime; // (added)
```

Struct proc에 프로세스가 할당된 CPU를 나타내는 cpu_affinity, 프로세스 누적 실행 시간을 기록하는 runtime 변수를 추가해줬다.

2. RR cycle: 10ms로 가정

```
p->runtime += 10; //Schduling cycle 당 10ms 가정 (added)
```

3. Big Core 이동

```
int runnable_count = 0; //(added)
  struct proc *heavy_proc = 0; //(added)
  for(p = ptable.proc; p < &ptable.proc[NPROC]; p++){//(added)
    if(p->state == RUNNABLE && p->cpu_affinity == LITTLE_CORE) {//(added)
      runnable_count++;//(added)
      if(!heavy_proc || p->runtime > heavy_proc->runtime) {//(added)
        heavy_proc = p;//(added)
      }
    }
  }

  // Migrate if over threshold
  if(runnable_count > 10 && heavy_proc) { // (added)
    migrate_process(heavy_proc); // (added)
  }
```

- Little Core의 RUNNABLE 상태 프로세스 수를 계산:
  - runnable_count 변수를 통해 Little Core에 할당된 RUNNABLE 상태의 프로세스

수를 count

- 만약 RUNNABLE 상태이고, cpu_affinity가 LITTLE_CORE인 프로세스가 있다면 runnable_count를 증가시킵니다.

- 가장 오래 실행된 프로세스를 찾음:
  - heavy_proc 변수를 사용하여 가장 오래 실행된 프로세스를 선택.
  - p->runtime을 기준으로 실행 시간이 가장 긴 프로세스를 heavy_proc에 저장
- Little Core의 부하가 임계치를 초과하면 프로세스를 이동:
  - runnable_count가 5보다 크면, 즉 Little Core의 부하가 높을 경우, migrate_process(heavy_proc)를 호출하여 가장 무거운 프로세스(가장 오래 실행된 프로세스)를 Big Core로 이동
  - Big Core 이동 함수: migrate_process
    선택된 프로세스를 Big Core로 이동

```
void
migrate_process(struct proc *p) { // (added)
    if (holding(&ptable.lock)) { // (added)
        if (p->cpu_affinity == LITTLE_CORE) { // (added)
            p->cpu_affinity = BIG_CORE; // (added)
        }
    } else { // (added)
        acquire(&ptable.lock); // (added)
        if (p->cpu_affinity == LITTLE_CORE) { // (added)
            p->cpu_affinity = BIG_CORE; // (added)
        }
        release(&ptable.lock); // (added)
    }
}
```

4. 결과
   - 프로세스 수: 총 24개의 프로세스가 생성.
   - Response time
     - 대부분 1. 5이상인 경우, 이전에 이미 많은 프로세스가 실행 중이어서 그런 것으로 분석됨.
     - Average response time: 2
   - Turnaround time
     - Average turnaround time: 555
   - Core 이동: 대부분 Little에서 Big으로 이동.

```
$ schedbench
1 (For project submission): Create 3 processes for each task 0-7
2 (For debugging): Create 1 process for each task 0-7
3 (Custom): Create N processes for each task 0-7
Select an option (1-3): 1
total_procs: 24
schedbench start
Time from fork() to wait(): 917
(0) [cpu 0->1] [tick 1048->1821] response time: 1, turaround time: 774
(1) [cpu 0->1] [tick 1049->1076] response time: 1, turaround time: 28
(2) [cpu 1->1] [tick 1049->1949] response time: 1, turaround time: 901
(3) [cpu 1->1] [tick 1049->1060] response time: 1, turaround time: 12
(4) [cpu 1->0] [tick 1049->1780] response time: 1, turaround time: 732
(5) [cpu 1->1] [tick 1050->1744] response time: 2, turaround time: 696
(6) [cpu 1->1] [tick 1051->1477] response time: 1, turaround time: 427
(7) [cpu 1->0] [tick 1051->1872] response time: 1, turaround time: 822
(8) [cpu 0->1] [tick 1053->1815] response time: 3, turaround time: 765
(9) [cpu 1->1] [tick 1053->1091] response time: 2, turaround time: 40
(10) [cpu 0->1] [tick 1055->1955] response time: 3, turaround time: 903
(11) [cpu 0->1] [tick 1055->1067] response time: 1, turaround time: 13
(12) [cpu 0->0] [tick 1055->1797] response time: 1, turaround time: 743
(13) [cpu 0->1] [tick 1058->1786] response time: 4, turaround time: 732
(14) [cpu 0->0] [tick 1058->1485] response time: 1, turaround time: 428
(15) [cpu 0->1] [tick 1058->1872] response time: 1, turaround time: 815
(16) [cpu 1->1] [tick 1058->1812] response time: 1, turaround time: 755
(17) [cpu 0->1] [tick 1058->1095] response time: 1, turaround time: 38
(18) [cpu 0->1] [tick 1064->1964] response time: 7, turaround time: 907
(19) [cpu 0->1] [tick 1064->1075] response time: 5, turaround time: 16
(20) [cpu 0->1] [tick 1064->1804] response time: 5, turaround time: 745
(21) [cpu 0->1] [tick 1064->1780] response time: 5, turaround time: 721
(22) [cpu 1->1] [tick 1064->1512] response time: 5, turaround time: 453
(23) [cpu 0->0] [tick 1065->1917] response time: 6, turaround time: 858
(Child) Total response time: 60 (avg: 2), Total turnaround time: 13324 (avg: 555)
schedbench end
```