

학과: 서어서문학과

학번: 2015131406

이름: 박가은

제출 날짜: 2020년 6월 10일

Freeday 사용 일수: 1일

과제 개요

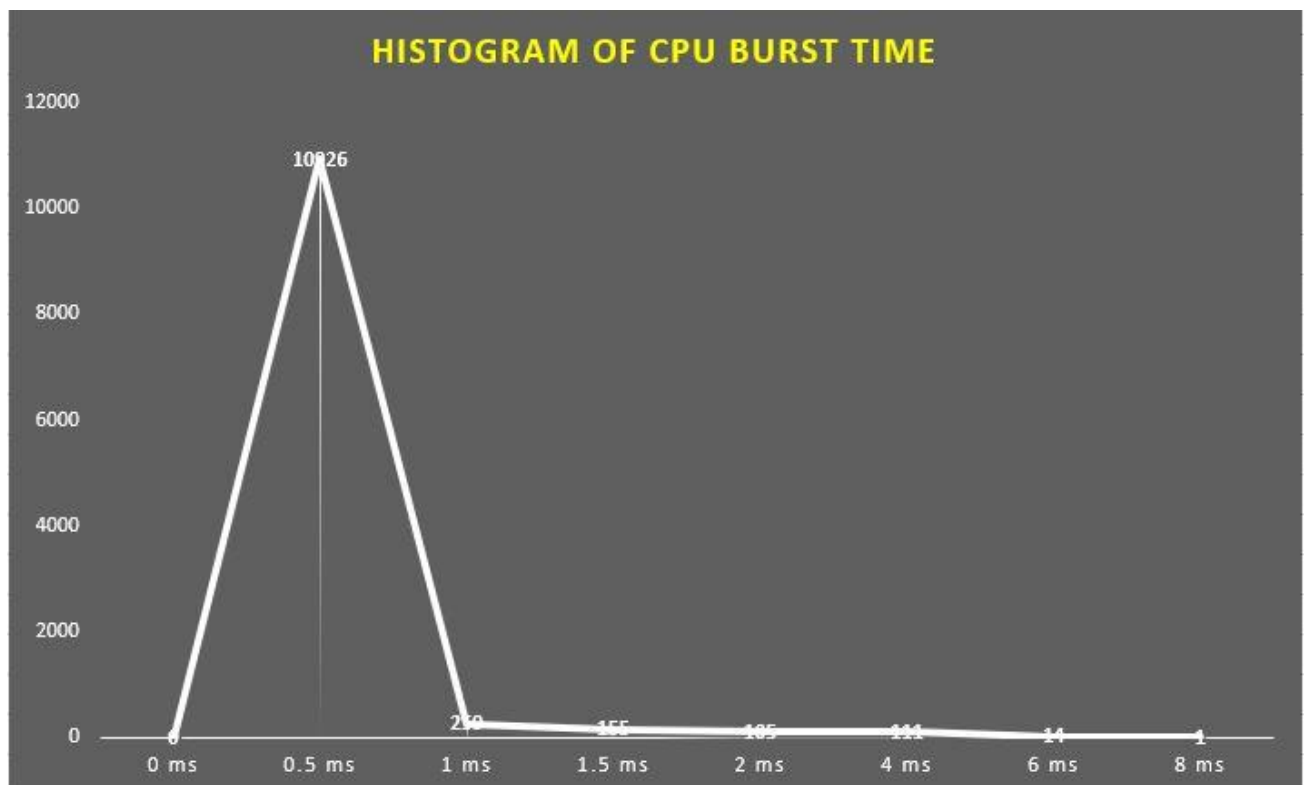
: CPU burst를 측정하기 위해 stats.h에 있는 sched_info_debt() 함수 안에 printk()함수를 이용하여 CPU Burst 값을 출력할 수 있도록 했다. 커널 프로세스 뿐만 아니라 우리가 일반적으로 사용하는 유저 어플리케이션의 프로세스의 CPU Burst도 측정하기 위해 우분투 안에 있는 firefox 웹 브라우저로 유튜브로 노래를 들으며 인터넷 쇼핑을 했다.

오늘날의 컴퓨터는 대부분 폰노이만 구조(von Neuman architecture)를 따른다. 즉, 모든 프로그램은 메모리에 올라와 실행할 수 있다는 뜻이다. 운영체제에서 프로세스란 하나의 작업 단위이다. 사용자가 프로그램을 실행하면 폰노이만 구조에 따라 디스크에 저장된 정적인 프로그램은 메모리에 올라와 동적인 프로세스가 된다.

스케줄러를 이해하기 위해서는 스케줄링을 알아야 한다. 스케줄링이란 여러 프로세스의 상황을 고려하여 CPU와 시스템 자원을 어떻게 배분할지 결정하는 일을 말한다. 스케줄러는 이러한 스케줄링을 수행하는 컴퓨터 자원을 의미한다.

CPU Burst에 대한 그래프 및 결과 분석

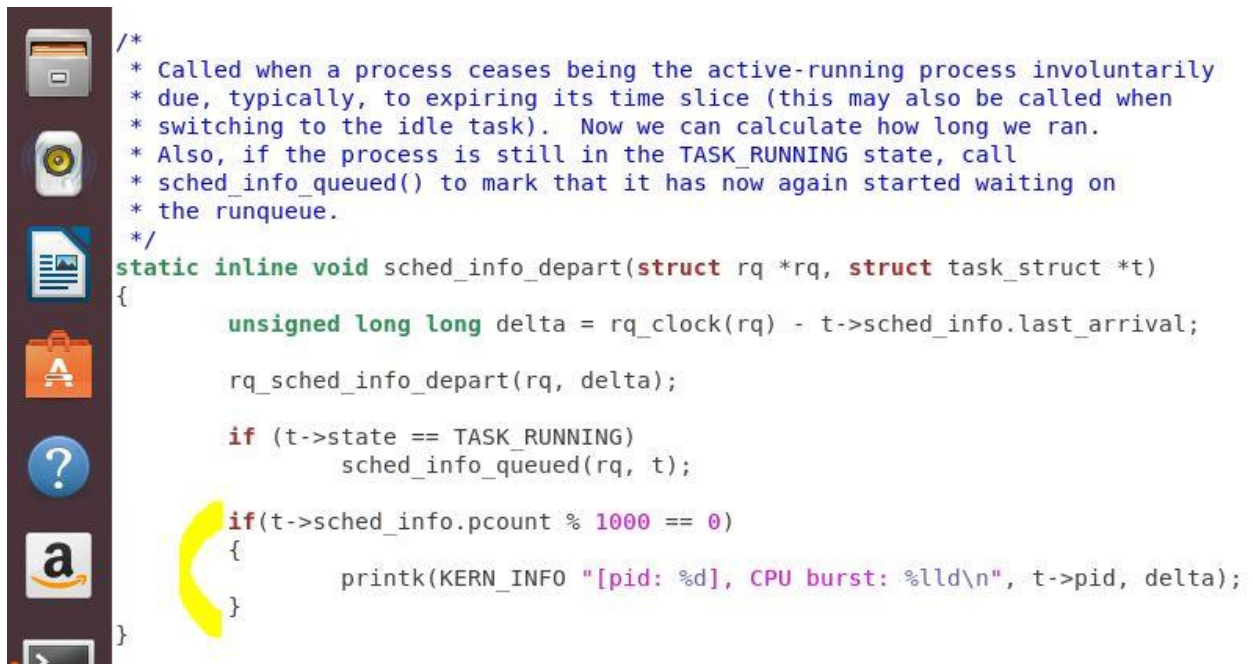
:



이 그래프에서 가로축은 CPU Burst의 duration이고, 세로축은 빈도수이다. 프로세스마다 다르지만 그래프를 보면 CPU Burst가 짧은 것이 압도적으로 빈도수가 많으며 대부분의 프로세스는 같은 CPU Burst duration을 가지고 있다는 뜻이다. CPU는 멀티 태스킹을 할 때 하나의 프로세스에 하나의 time slice를 준다. 만약 프로세스가 자신에게 할당받는 타임 슬라이스가 끝나면, 혹은 I/O를 대기해야 한다면, 혹은 프로세스가 자신의 일을 모두 끝마쳤다면 CPU에서 내려와야 하기 때문이다. 모든 프로세스는 같은 time slice를 공정하게 할당받으므로 프로세스의 CPU Burst duration은 비슷한 값을 가질 수 밖에 없는 것이다.

코드 분석

:



```
/*
 * Called when a process ceases being the active-running process involuntarily
 * due, typically, to expiring its time slice (this may also be called when
 * switching to the idle task). Now we can calculate how long we ran.
 * Also, if the process is still in the TASK_RUNNING state, call
 * sched_info_queued() to mark that it has now again started waiting on
 * the runqueue.
 */
static inline void sched_info_depart(struct rq *rq, struct task_struct *t)
{
    unsigned long long delta = rq_clock(rq) - t->sched_info.last_arrival;
    rq_sched_info_depart(rq, delta);

    if (t->state == TASK_RUNNING)
        sched_info_queued(rq, t);

    if(t->sched_info.pcount % 1000 == 0)
    {
        printk(KERN_INFO "[pid: %d], CPU burst: %lld\n", t->pid, delta);
    }
}
```

stats.h에 있는 sched_info_depart 함수에 CPU Burst를 출력할 수 있도록 코드를 수정했다. t는 task_struct 구조체를 가리킨다. task_struct에는 프로세스의 정보를 저장하고 있는데 그 중 sched_info.pcount에서는 CPU에 얼마나 많이 올라갔는지 그 빈도수를 의미한다. 그래서 1000번 볼릴 때마다, 즉 t->sched_info.pcount % 1000 == 0일때마다 CPU Burst 값을 출력하도록 했다. CPU Burst 값은 sched_info_depart에 선언되어 있는 delta 변수를 사용했다. delta는 rq_clock(rq), 현재 시각에서 sched_info.last_arrival, CPU에 올라온 시각을 뺀 값이다. 현재 CPU에서 해당 프로세스가 사용한 CPU Burst값을 담고 있는 것이다. 프로세스의 pid도 또한 task_struct에 저장되어 있기 때문에 t->pid로 하여금 프로세스 아이디도 출력할 수 있도록 했다.

문제점과 해결 방법

: 큰 문제점은 없었다. 다만, CPU Burst 값으로 착각하여 vruntime을 출력한 적이 있었다. 하지만 vruntime은 리눅스가 CPU 스케줄링 할 때 프로세스에게 부여한 우선순위도 더한 값이다. 때문에 vruntime을 CPU Burst로 출력하는 것은 오류가 있다고 판단하여 sched_info_depart()에 있는 변수 delta를 사용하여 CPU Burst를 출력하였다.