**2020년 1학기 운영체제 2차 과제**

**학과** 컴퓨터학과

**학번** 2018320112

**이름** 김현아

**제출일** 2020.6.13

**Freeday 사용 일수** 4일

**목차**  과제 개요

프로세스와 스케줄러의 개념

CPU burst 그래프 및 결과분석

소스코드 설명

과제 수행 시의 문제점 및 해결방안

1. **과제 개요**

과제는 리눅스 스케줄러를 수정하여 CPU burst를 측정하는 것이었다. burst값이 1000회 호출시 1회 기록하도록 조건이 주어졌으며, 측정된 log들을 그래프로 변환하여 알려진 CPU 점유 시간과 비교하는 것으로 마무리된다.

이 과제를 위해 1. 변수 찾기 및 함수 작성 2. 컴파일 3. 로그 작성 및 출력 4. 그래프 작성 5. 보고서 작성의 단계로 진행하였고, linux-4.20.11 환경에서 진행하였다. 또한 그래프 작성은 python의 matplotlib를 이용하였다. 또한 제시된 그래프와 비슷한 꺾은선 그래프를 작성할 때는 spss를 사용하였다.

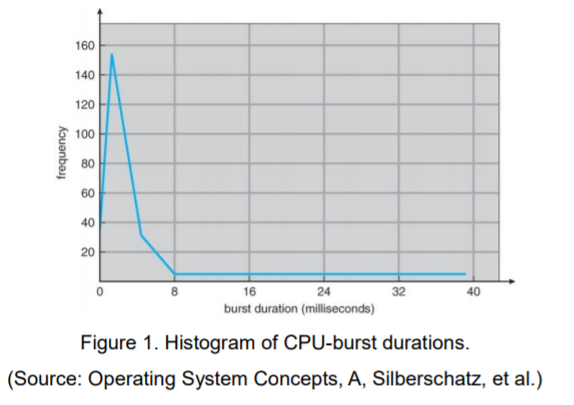
30분간 측정 시간동안 기본적으로 파이어폭스를 이용해 유튜브 영상을 재생 시키고 있었으며, 네이버 블로그와 스토쿠, 계산기 등의 프로그램을 실행시켰다.

1. **프로세스와 스케줄러의 개념**

OS에서 프로세스는 source code가 일련의 과정을 거쳐 memory에 load됨으로써 만들어지는, 스케줄링의 단위이다. 이 때 스케줄링이란 어떻게 프로세스에 CPU의 사용을 분배할 것인가에 관한 문제이며, 스케줄러가 memory에 load된 프로세스 가운데 하나를 선택하여 CPU를 할당하게 된다. 이 때 스케줄링의 최대 목표는 bottleneck을 줄여 CPU를 최대한 활용하는 것이다. 이 때, CPU 사용률, 처리량, 응답/대기시간 등 여러 조건을 만족하는 것이 가장 이상적인 스케줄러이나, 그것은 현실적으로 불가능하기에 각 시스템의 policy에 맞추어 스케줄링을 진행하게 된다.

스케줄링 된 프로세스의 수행 사이클은 CPU burst(CPU 연산을 수행하는 시간)과 I/O Burst(I/O 처리를 위해 기다리는 시간)이 번갈아 가며 수행하게 된다. 이 때, CPU burst time은 프로세스, 시스템의 종류를 막론하고 대개 아래와 같은 형태를 가지게 된다.

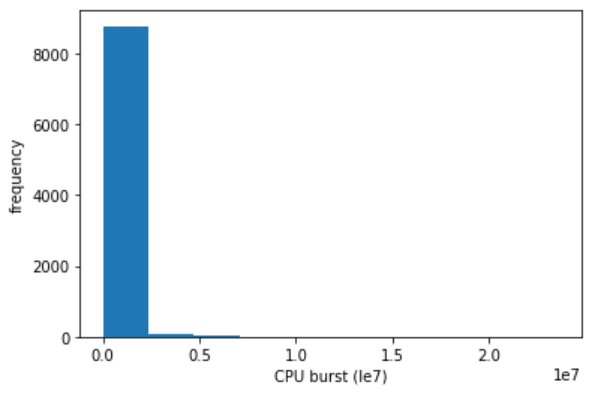
이 때 CPU burst는 특정 프로세스가 CPU를 점유한 시간을 의미한다. 아래의 히스토그램을 통하여 time quantum을 설정하는데, 아래의 히스토그램을 기준으로 설명하면, burst의 대부분이 8ms이하인 것을 확인할 수 있다. 대부분의 process들이 time quantum 안에 끝나도록 하여 overhead를 줄이는 것이 가장 바람직하다.



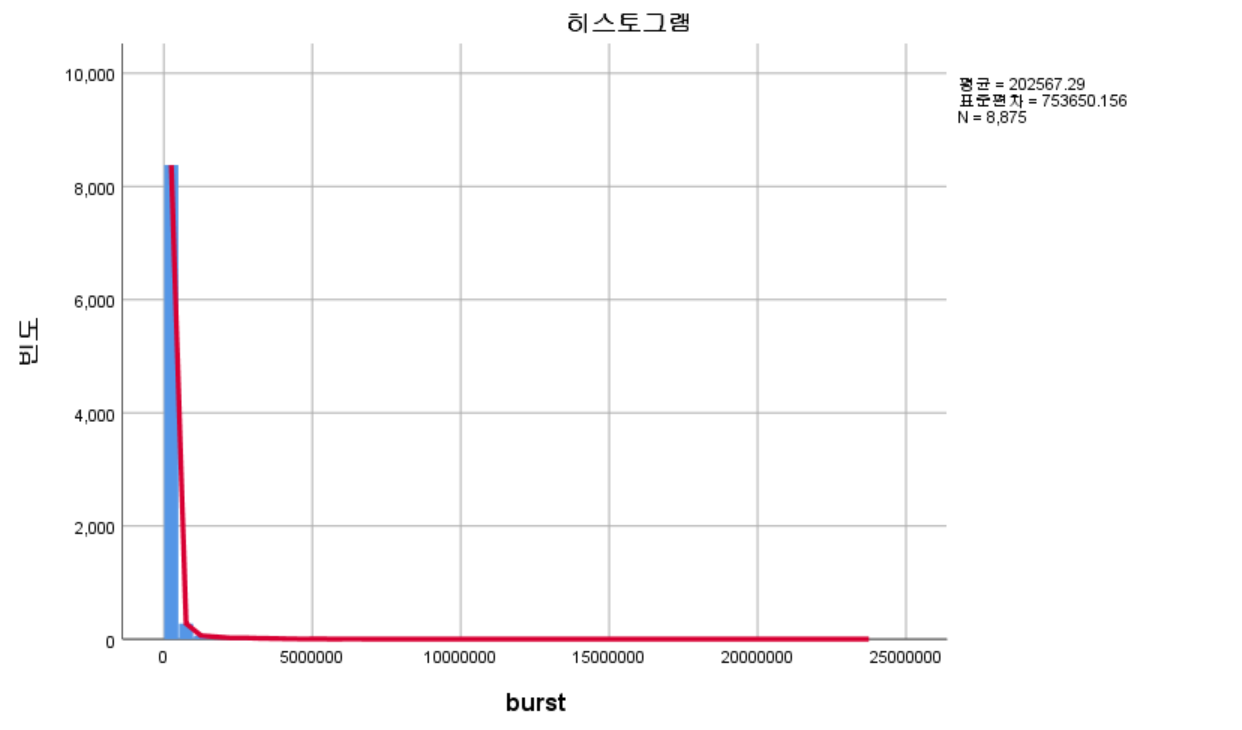
이 스케줄링을 시행할 때 FCFS, SJF 등 다양한 알고리즘을 사용할 수 있는데, 본 과제에서는 linux 환경에서 사용되는 스케줄링 방식인 CFS(Completely fair scheduler) 방식을 사용한다.

CFS의 특징은 공정성으로, 실행 대기 상태인 프로세스들을 우선 순위에 따라 공정하게 실행하는 스케줄러이다. 다르게 말하자면, 성능과 공평성 간의 균형을 중요시한다. 가상의 CPU 사용시간인 vruntime에 따라 스케줄링 순서를 결정하며, 이 방식의 이점은 오랫동안 미실행된 프로그램의 vruntime이 상대적으로 적기 때문에 starvation을 방지할 수 있다는 것이다. 가상 CPU 사용 시간인 vruntime은 가중치가 부여된 CPU 시간을 의미하며, 실제 CPU 사용시간보다 우선시하는 잣대가 된다. 그러나 우선순위나 CPU 사용시간이 vruntime에 반영되므로, 실제 사용시간이나 우선순위가 무시되는 것은 아니다.

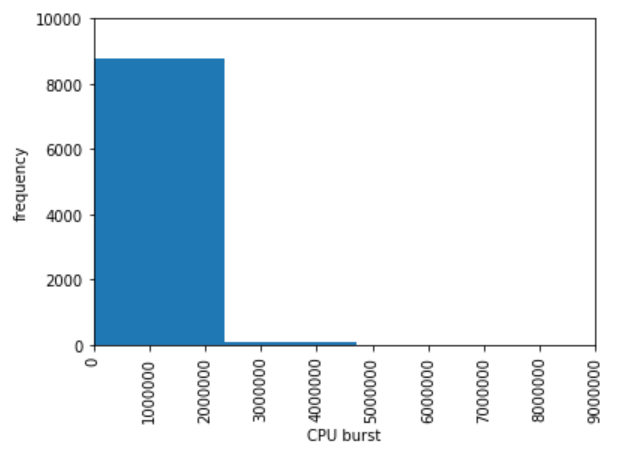
1. **CPU burst 그래프 및 결과분석**



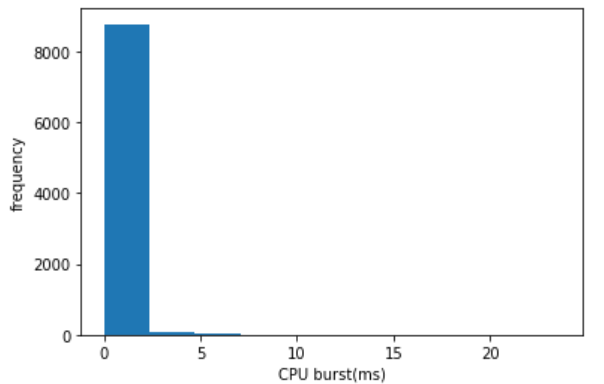
위 그래프는 전체 process의 CPU burst를 측정하여 그래프로 그린 그림으로, 자연로그를 취해 가장 보기 좋은 형태로 그려졌다. 항목2의 <Histogram of CPU-burst durations>과 비슷한 형태를 가지고 있음을 확인할 수 있다.



위 그래프는 이전의 그래프를 과제 조건에 맞게 꺾은 선 그래프로 작성한 것이다.

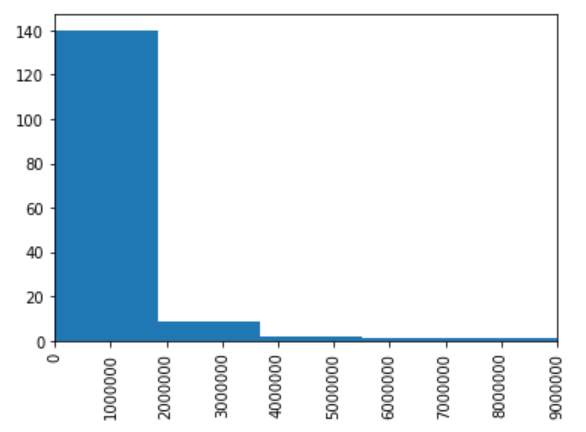


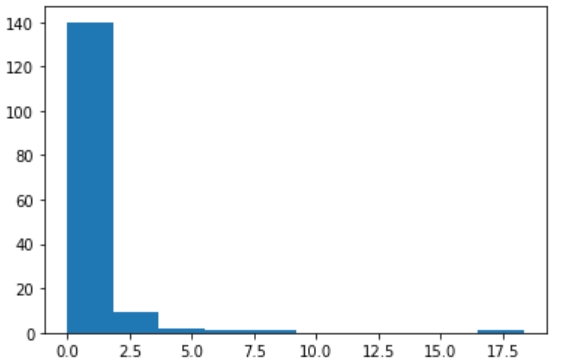
위 그래프는 전체 프로세스의 cpu burst와 frequency 모두 실제 값을 반영하여 그린 그래프이다.



위 그래프는 전체 프로세스의 cpu burst를 ms단위로 변환하여 작성한 그래프이다. Burst의 대부분이 5ms이하인 것을 확인할 수 있다. 따라서 이 CPU의 경우 2.5ms 또는 5ms이하로 time quantum을 설정하는 것이 바람직하다고 생각한다.

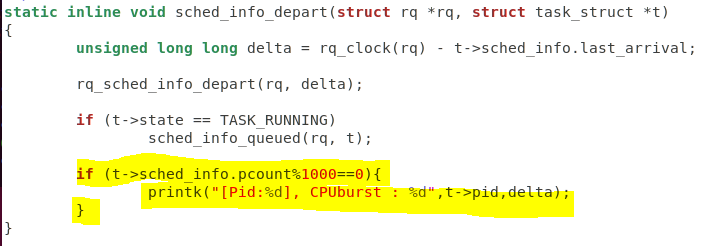
끝으로, 특정 프로세스에 대한 그래프 작성 역시 진행하여 보았는데,





위의 두개 그래프들은 pid 1330에 대한(특정 프로세스에 대한) 그래프로 첫 번째 그래프는 실제 값을 반영한 그래프, 두번째는 밀리초 단위로 x축을 설정한 그래프이다. 이를 통해 특정 프로세스나 전체 프로세스를 대상으로 측정했을 때 모두, 비슷한 형태를 가짐을 확인할 수 있다.

1. **소스코드 설명**



/usr/src/linux-4.20.11/kernel/sched/stats.h 파일의 sched\_info\_depart 함수의 형광펜 부분을 추가하였다. 사용한 변수는 pcount, pid, delta 세 개이다. 우선 pcount는 burst값을 측정하여 1000회 호출시 1회 기록되도록 조건을 설정하였다. Delta는 현재 시간-프로세스가 CPU 점유를 시작했을 때의 시간(ns)을 의미한다.

소스코드 작성 과정은 다음과 같다. 우선 stats.h 파일 안의 sched\_info\_depart 함수를 분석하여 어떤 코드를 작성해야 1000번에 한번 pid와 CPU burst time을 출력하는지 확인한다. 다음으로 tast\_struct를 분석하여 어떤 변수가 프로세스마다 1000회 호출되는지 그 값을 담고 있는지를 확인한다. 그 후, 조건에 맞는 코드를 작성하였다.

1. **과제 수행 시의 문제점과 해결 과정 또는 의문 사항**

Dmesg를 통해 log를 출력하여 txt파일로 바꾸는 과정에서 양이 너무 많아서인지 30분치 측정량 중 앞의 10여분이 날아갔다. 그 후 재측정을 실시하였는데, 이 때는 10분에 한 번씩 txt파일을 만들었고, 그래프를 그리기 전 로그들을 다 합쳐 사용했다.

그래프를 그리는 과정에서 CPU burst를 process별로 그리는 것으로 오해하고 각 pid마다그래프를 작성하였다. 다시 2차 과제 설명서를 읽어 보니 전체 프로세스에 대한 CPU burst time만을 기록하면 되는 것임을 확인하였다.

이번 과제 수행 중에서는 코드 작성이 한 번에 오류 없이 끝나서 큰 어려움 없이 마무리할 수 있었던 것 같다.