로고이(가) 표시된 사진

자동 생성된 설명

운영체제

Assignment 5 과제

수업 명 : 운영체제

과제 이름 : assignment5

담당 교수님 : 최상호 교수님

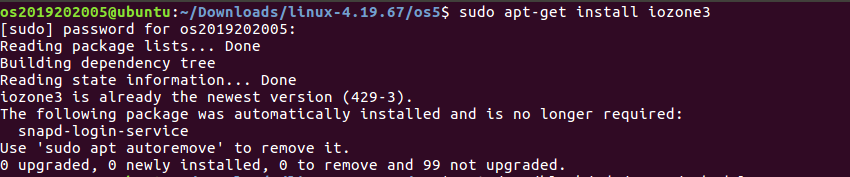
학 번 : 2019202005

이 름 : 남종식

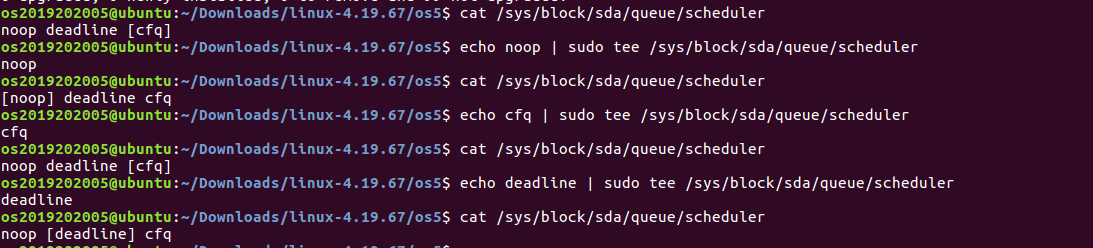
**Introduction**

이번 과제는 I/O Zone을 이용한 세 가지 Linux I/O scheduler 성능 테스트를 하는 과제입니다. 세가지 Linux I/O scheduler는 noop, deadline, CFQ등이 있습니다.  
noop은 매우 간단한 I/O 스케줄링 알고리즘으로, 큐에 들어온 I/O 요청을 순서대로 처리합니다. 디스크 I/O 요청을 큐에 넣으면 가장 먼저 들어온 요청부터 처리하므로, 순차적인 디스크 I/O에는 적합합니다.   
Deadline은 각 I/O 요청에 대해 deadline을 부여하고, 이 마감 시간 안에 요청이 처리되도록 스케줄링합니다. 디스크 I/O의 응답 시간을 개선하고, 실시간 응용 프로그램에 대한 보장된 응답 시간을 제공하는 데 중점을 둡니다.  
CFQ는 프로세스에 대한 공정성을 중시하는 I/O 스케줄러로, 각 프로세스에 대해 디스크 I/O를 공정하게 분배하려고 노력합니다. 디스크 대역폭을 동적으로 조절하고, 여러 프로세스 간에 I/O 대역폭을 분배하여 응용 프로그램의 반응성을 향상시키기 위해 설계되었습니다.  
실험 시에는 record size를 변경해가면서 실행 후 이에 대한 결과를 비교하고 이를 분석해야합니다.

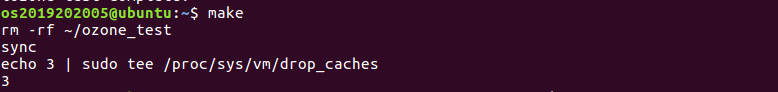
**Conclusion & Analysis**



IOZone설치를 진행하였습니다.



Scheduler를 확인하고 변경하는 과정을 진행하였습니다.



캐시 및 버퍼를 비워서 실험에 영향을 주는 요소를 제거하기 위하여 이를 makefile로 만들어 매 실험 전에 진행하였습니다.













****



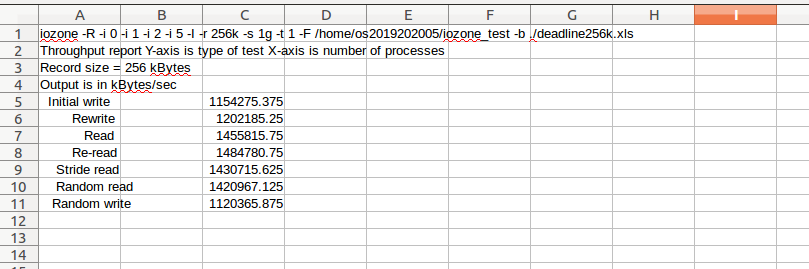


먼저 -R을 통해 excel report를 생성하였고 -b[filename]을 통해 excel과 호환 가능한 binary file을 생성하였습니다. -i를 통해 write/re-write read/re-read random-read/write 그리고 strid-read를 따로 선택하여 진행했습니다.

Stride -read를 선택한 이유는 일반적인 write와 read를 제외하고 random read/write 측정이 있어 이와 대비되는 작업을 진행하고 싶어 선택하였습니다. Random read는 임의의 위치에서 읽는 반면, stride read는 일정한 간격으로 건너뛰어 데이터를 읽습니다. 이러한 이유로 stride read를 선택하게 되었습니다.

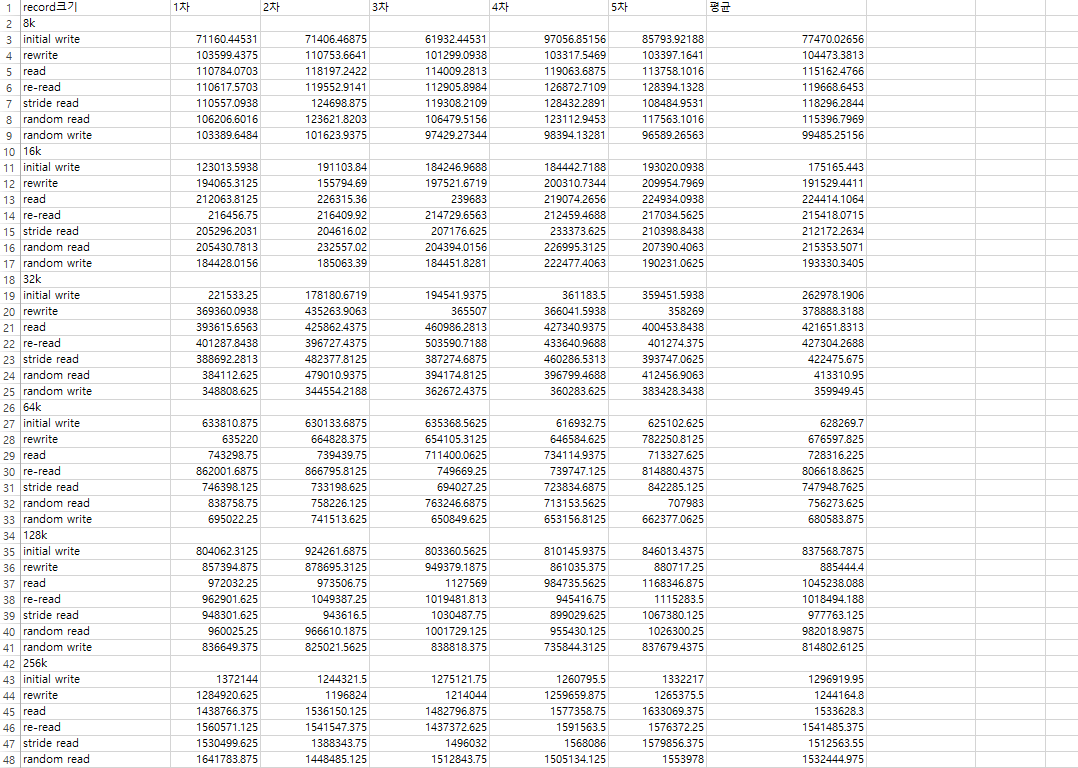
-I를 통해 파일에 대한 모든 operation은 buffer cache를 우회 및 disk 직접 이동하게 했습니다. -r을 통해 record를 직접 설정하여 이를 바꿔주며 테스트했습니다. -s를 통해 file size를 1 GB로 설정하였고 -t를 통해 thread를 1개만 사용하였습니다. -F를 통해 thread or process의 파일 경로를 ~/iozone\_test으로 설정하였습니다.

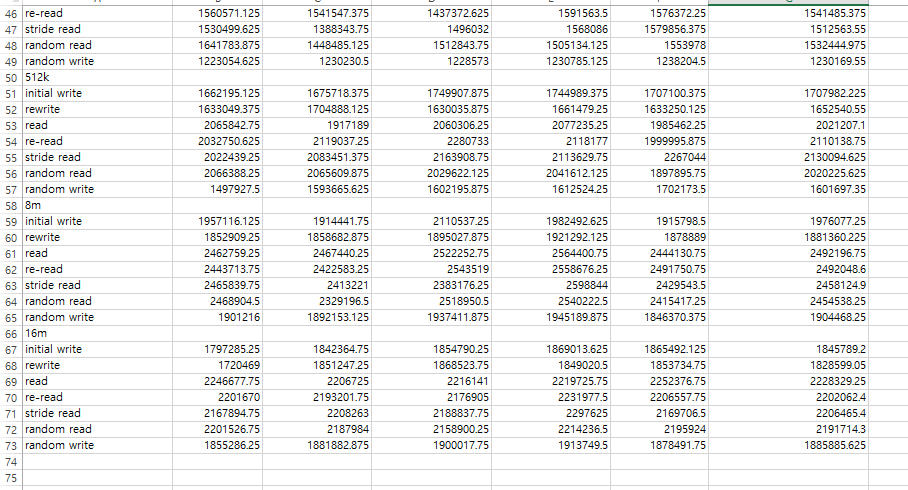
위 9번의 과정을 5번씩 총 3개의 scheduler에서 진행하였습니다.



위 명령어를 진행하면 다음과 같은 excel파일이 생성되는 것을 확인할 수 있으며 이 출력된 결과들을 다른 excel파일에 정리하여 평균 값을 구했습니다.

**Scheduler 별 성능 결과 분석**





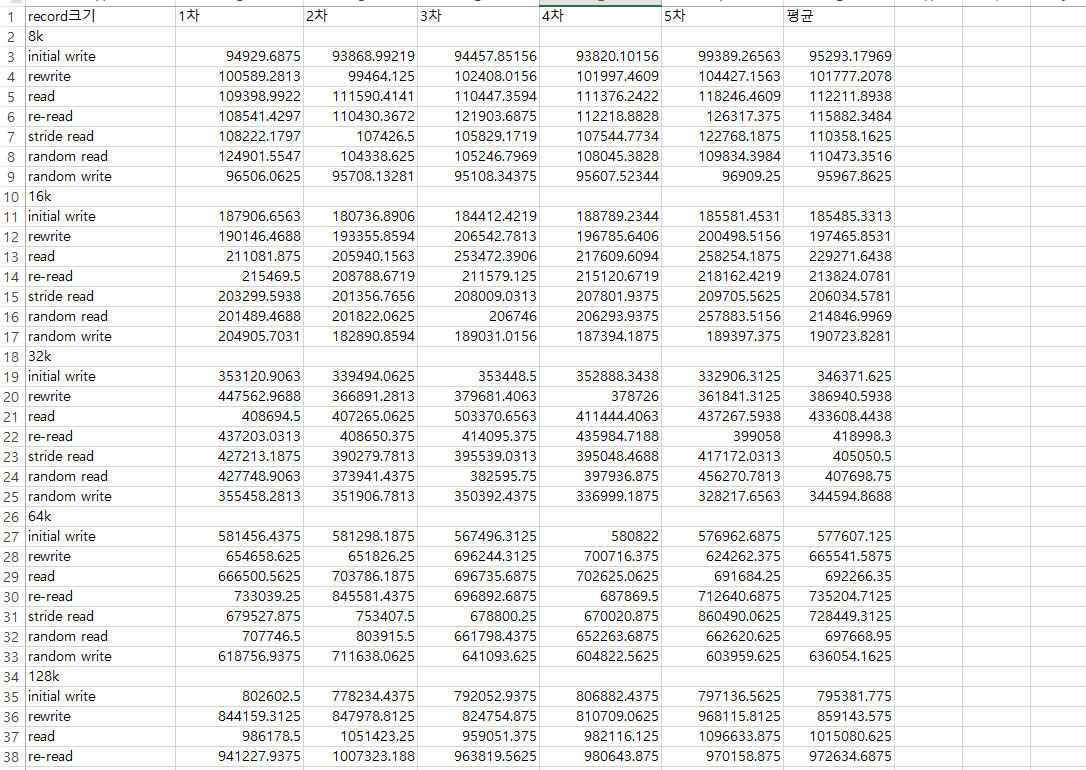
위 화면은 noop scheduler를 사용해 측정한 값입니다.

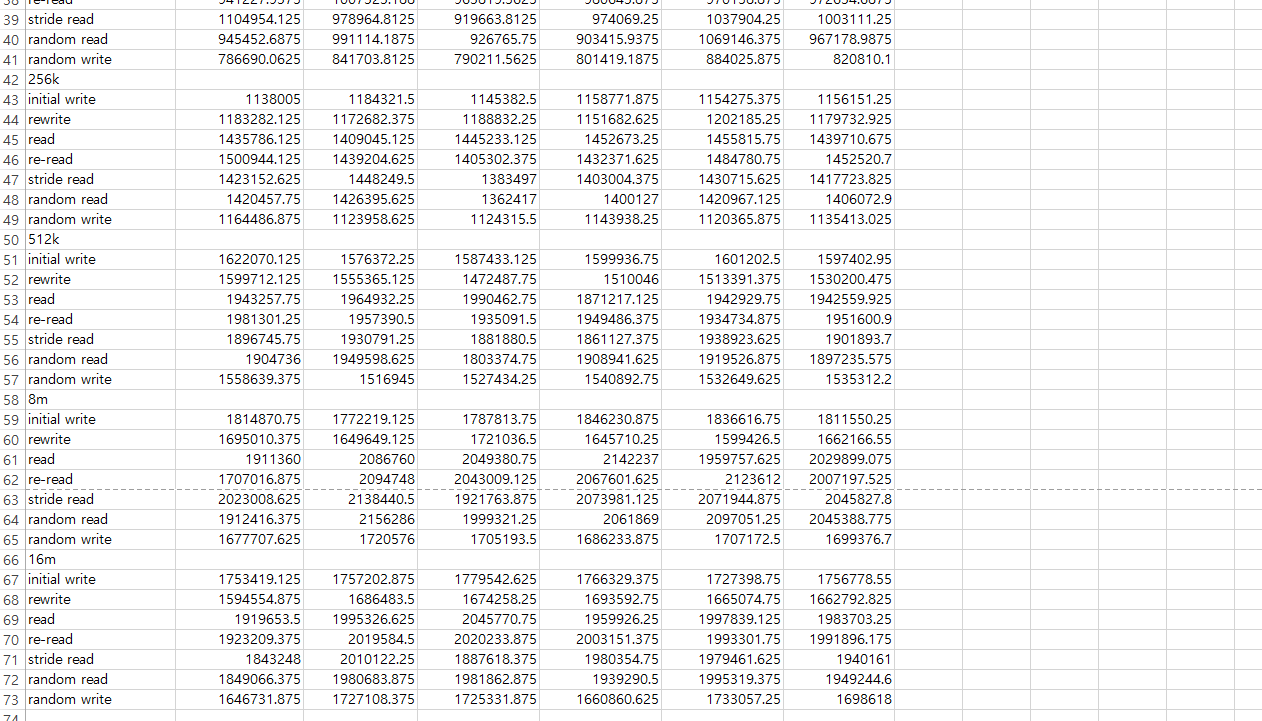
텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

위 화면은 noop scheduler의 실험을 5회 이상 실시하여 얻은 평균값을 바탕으로 그래프를 나타낸 것입니다.

먼저 왼쪽부터 7개씩 여백을 기준으로 8KB / 16KB / 32KB / 64KB / 128KB / 256KB /512KB / 8MB / 16MB의 record size를 변경하여 진행 후 얻은 평균 값입니다. 여백으로 분리된 7개의 그래프는 순서대로 initial write, rewrite, read, re-read, stride read, random read, random write의 평균 값입니다. 먼저 record size가 커질수록 대체로 성능이 좋은 모습을 확인할 수 있었으며 8M일때는 16M일 때 보다 좋은 성능을 보입니다. Noop scheduler는 Random access를 하는 device를 위한 scheduler이기 때문에 re-read보다 더 좋은 결과를 내는 경우도 발생하는 모습을 볼 수 있습니다.



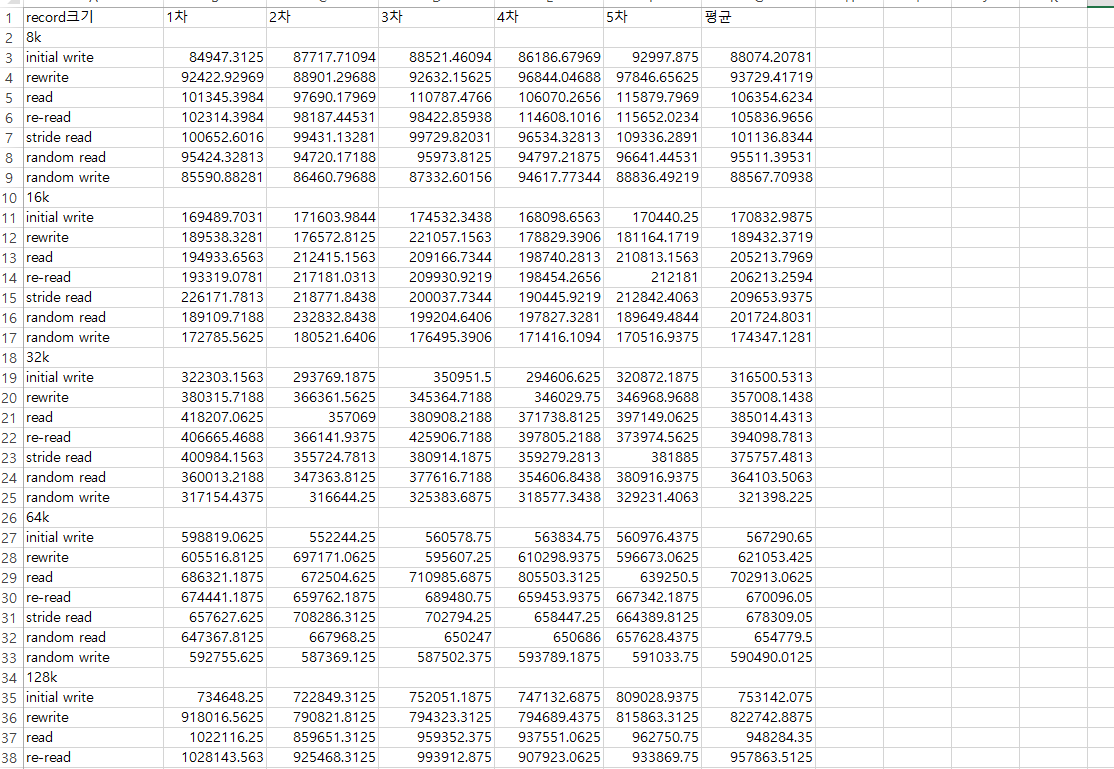


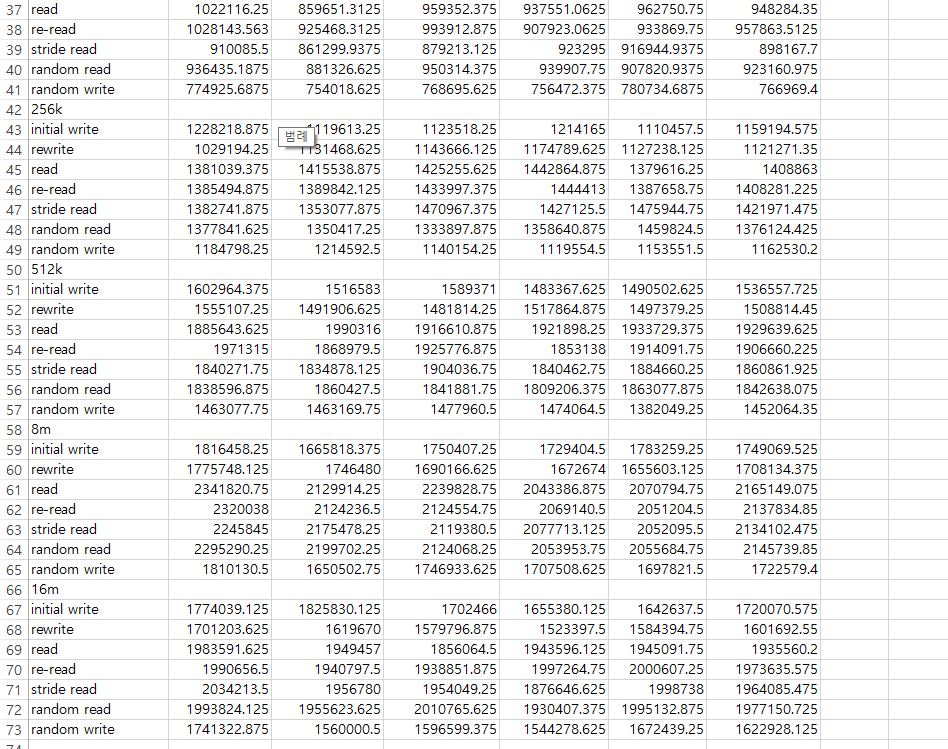
위 화면은 deadline scheduler을 사용해 측정한 값입니다.

텍스트, 스크린샷, 그래프, 라인이(가) 표시된 사진

자동 생성된 설명

위 화면은 deadline scheduler의 실험을 5회 이상 실시하여 얻은 평균값을 바탕으로 그래프를 나타낸 것입니다. Record size가 증가함에 따라 성능이 대체로 좋아지는 모습을 이 그래프에서도 확인할 수 있었으며 deadline scheduler는 512K부터 거의 성능이 크게 증가하지 않고 거의 비슷한 모습을 보였습니다.





위 화면은 cfq scheduler을 사용해 측정한 값입니다.

스크린샷, 텍스트, 도표, 그래프이(가) 표시된 사진

자동 생성된 설명

위 화면은 cfq scheduler의 실험을 5회 이상 실시하여 얻은 평균값을 바탕으로 그래프를 나타낸 것입니다. Cfq scheduler또한 record size가 증가함에 따라 성능이 대체로 좋아지며 이는 입출력을 요청하는 모든 프로세스들에 대해 디스크 I/O 대역폭을 공평하게 할당하는 것을 보장하는 기법이므로 record size가 작을 때에는 성능 면에서 크게 차이 나지 않는 모습을 확인할 수 있습니다. 하지만 record size가 커질수록 차이가 조금씩 나는 모습 또한 확인할 수 있습니다. 기존에 cfq scheduler의 이론적 정의와는 다른 모습을 보이는 측면도 있었습니다. 공평하게 할당하여 오차범위가 비슷할 것 같다고 생각했지만 오차가 발생하는 모습이었습니다. 그리고 8M일 때 16M일 때 보다 성능이 좀 더 좋은 모습도 확인할 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

8KB : 대부분의 테스트 연산에서 noop이 제일 좋은 성능을 보입니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

16 KB : 이전보다 오차가 크게 차이 나지 않는 것을 확인할 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

32KB : record size가 커질수록 deadline의 성능이 좋아지는 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

64KB : noop에서 random accecss에서의 성능이 점점 좋아지는 것을 확인할 수 있으면 cfq가 deadline보다 성능이 더 좋은 경우도 볼 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

128KB : 큰 차이 없이 이전과 비슷한 성능을 확인할 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

256KB : deadline과 cfq의 성능이 좋아지는 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

512KB : 세개의 scheduler의 성능이 비슷해지는 모습을 확인할 수 있습니다. 하지만 noop이 가장 성능이 좋은 모습을 볼 수 있습니다.

텍스트, 스크린샷, 평행, 라인이(가) 표시된 사진

자동 생성된 설명

8MB : 대부분 cfq가 deadline보다 성능이 좋은 모습을 확인할 수 있습니다.

텍스트, 스크린샷, 라인, 평행이(가) 표시된 사진

자동 생성된 설명

16MB : deadline과 cfq가 성능이 비슷하고 noop이 이들보다 성능이 좋은 모습을 확인할 수 있습니다.

**고찰**

이번 과제에서는 I/O Zone을 이용한 세 가지 Linux I/O scheduler 성능 테스트를 진행하였는데 scheduler 3개를 가지고 9개의 record size 그리고 이를 5번씩 진행하여 135번 진행하였습니다. 처음에 과제를 시작할 때는 조금 막막하였지만 하다 보니 생각보다 빠르게 진행할 수 있었습니다. 과제를 하면서는 굳이 이렇게 많이 테스트할 이유가 있나 라는 생각도 들었지만 한편으로는 어떠한 실험을 하여 결과를 얻기 위해서는 이렇게 많은 데이터가 필요하구나 라는 생각을 했습니다. 더 나아가 이런 과제 수준의 실험이 아니라 정말 연구실에서 진행하는 실험에는 얼마나 많은 데이터 셋이 필요할까 라는 생각도 들었습니다.  
실험적인 부분에서는 이번 실습 강의자료에서 처음 본거라 낯설었지만 이론적인 부분에서는 운영체제 강의시간에 다 배운 내용이어서 크게 어려운 부분은 없었습니다. 이번 실험에서는 특히 캐시 버퍼를 지우는 것이 굉장히 중요한 것 같았다. 이 과정을 진행하지 않으면 실험 결과가 크게 달라질 것이고 이러한 결과는 의미가 없기 때문에 캐시 버퍼를 지우는 과정을 빼먹지 않으려 노력했습니다. 하지만 오차를 발생시키는 요소가 다른 부분에서 발생한 것 같았습니다. 예상과 다른 결과가 좀 있었는데 이러한 요소들을 완전히 제거하지 못한 부분들이 조금 아쉬웠습니다.

**Reference**

2023년 2학기 운영체제실습 13주차 I/O Scheduler

2023년 2학기 운영체제 & 운영체제실습 Assignment 5