System Programming Project 5

담당 교수 : 김영재

이름 : 전민수

학번 : 20171687

1. **개발 목표**

이번 프로젝트에서는 미니 주식 서버를 구현한다.

한명의 유저 뿐 아니라, 다수의 유저가 동시에 접속 요청을 했을 때에도 접속해서 동시에 처리가 가능한 서버를 구현하고자 한다.

2. **개발 범위 및 내용**

A. 개발 범위

1. select

이벤트 기반 서버를 구현한다. 이벤트를 기반으로 하여 여러 사용자가 들어왔을 때, select문을 통해 이번 loop에서 처리해주어야 하는 친구들을 모두 처리해주는 방식을 사용하였다.

2. pthread

쓰레드 기반 서버를 구현한다. 일정 쓰레드를 열어두고, 쓰레드와 사용자를 1대1로 매핑을 해줌으로써 처리를 해주는 서버를 구현하였다.

B. 개발 내용

- **아래 항목의 내용만 서술**
- **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
- **select**
  - ✓ select 함수로 구현한 부분에 대해서 간략히 설명

    fdmax를 통해 우리가 받았던 client 중 가장 큰 숫자의 클라이언트를 fdmax에 저장한다.

    fdmax만큼 루프를 돌면서 이미 있던 클라이언트의 경우에는 이미 select에 등록이 되어있기 때문에, 한번의 입력을 받아주고,

    없던 클라이언트의 경우에는 없는 클라이언트를 위한 fd인 listenfd에 붙어서 새로운 등록 절차를 거친 후에 이벤트 처리가 된다.
  - ✓ stock info에 대한 file contents를 memory로 올린 방법 설명

    프로그램이 시작할 때 file contents를 우선 메모리에 올린다. 이후, 메모리의 값

과 파일의 값을 동기화를 해주어야하는데, 이는 현재 우리의 이벤트에 등록된 자식이 0 명일때에만 그것을 진행한다.

- **pthread**

  ✓ pthread로 구현한 부분에 대해서 간략히 설명

  쓰레드는 readers writers problem과 producer consumer problem이 모두 생기 수 있다. 우선 클라이언트가 서버에 붙는 과정에서, thread가 다 차면, 클라이언트가 더 붙을 수 없기 때문에 producer consumer problem이 생긴다.

  또한, 각각의 show, buy, sell의 과정에서, show는 reader, buy와 sell은 writer라고 할 수 있기 때문에, readers writers problem이 생긴다.

  각각의 문제에 대해 mutex락을 걸어주었고, readers writers problem에 대해서는 이진트리의 각 노드에 대해서 mutex락을 걸어줌으로써 성능이 줄어드는 현상을 최대한 막고자 하였다.

C. 개발 방법

- **B.의 개발 내용을 구현하기 위해 어느 소스코드에 어떤 요소를 추가 또는 수정할 것인지 설명. (함수, 구조체 등의 구현이나 수정을 서술)**

- 기본적인 buy, show, sell의 구현을 위해서는 binary tree를 만들어야한다.

  때문에 binary tree의 node를 구성하는 구조체를 하나 구현해야하고, 그것에 따른 push함수, free함수를 각각 구현한다. 이후, buy show sell을 구현해서 기본적인 주식 서버의 기능을 갖춘다.

  쓰레딩으로 인한 문제를 해결하기 위해선 각각의 binary tree 노드 별로 reader mutex와 writer mutex를 들고 있는 편이 좋다.

- select

  기본으로 주어진 서버에서, select를 이용하여 이벤트 기반으로 바꿔준 이후에에도, echo 함수를 수정할 필요가 있다. while을 돌면서 클라이언트가 어떤 입력을 주나 기다리면 하나의 클라이언트에 대해서만 동작을 할 수밖에 없으므로, while

이 아닌 if를 두고, select를 통해 계속 echo 함수를 방문하는 식으로 구현한다.

- pthread

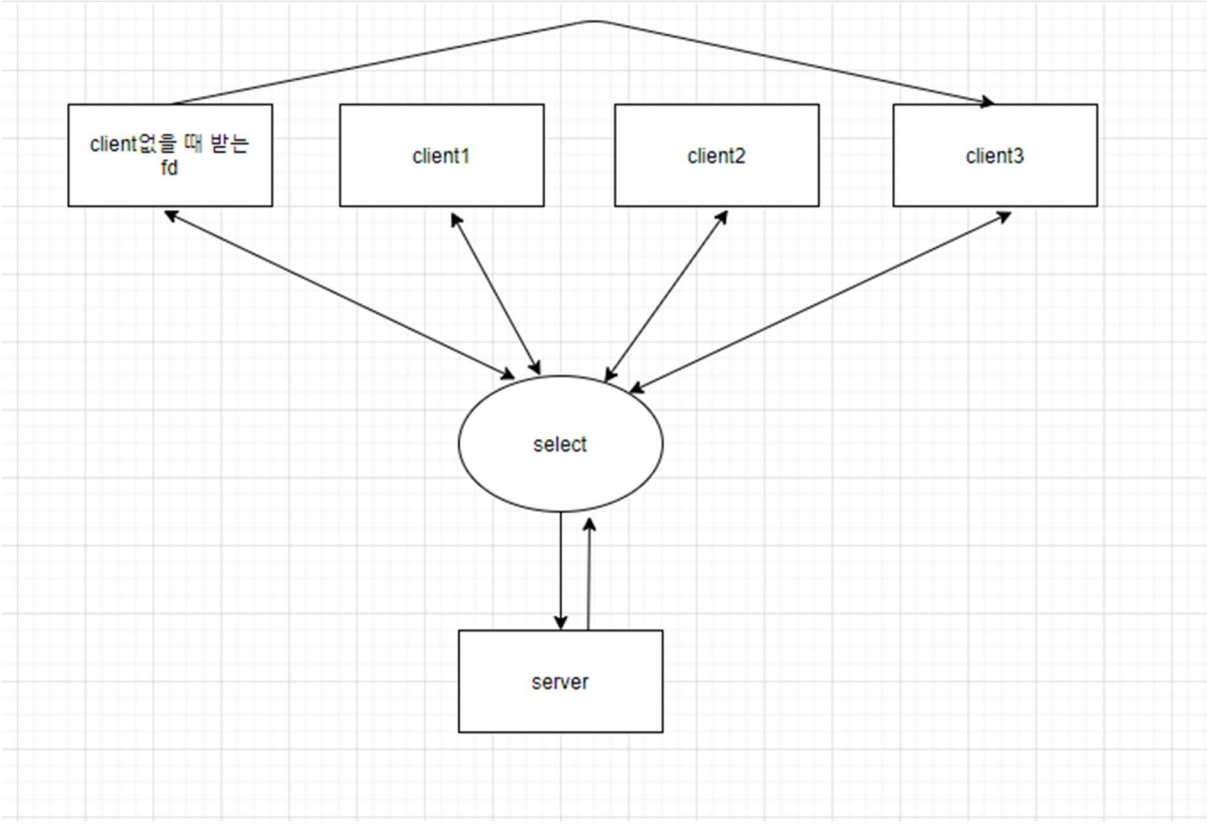
쓰레드의 problem들을 해결하기 위해 이진트리에 접근하는 함수들에 mutex락을 걸어줘야한다. 또한, 자식이 나의 쓰레드보다 많이 올 경우를 대비한 다른 mutex를 필요로한다.

3. 구현 결과

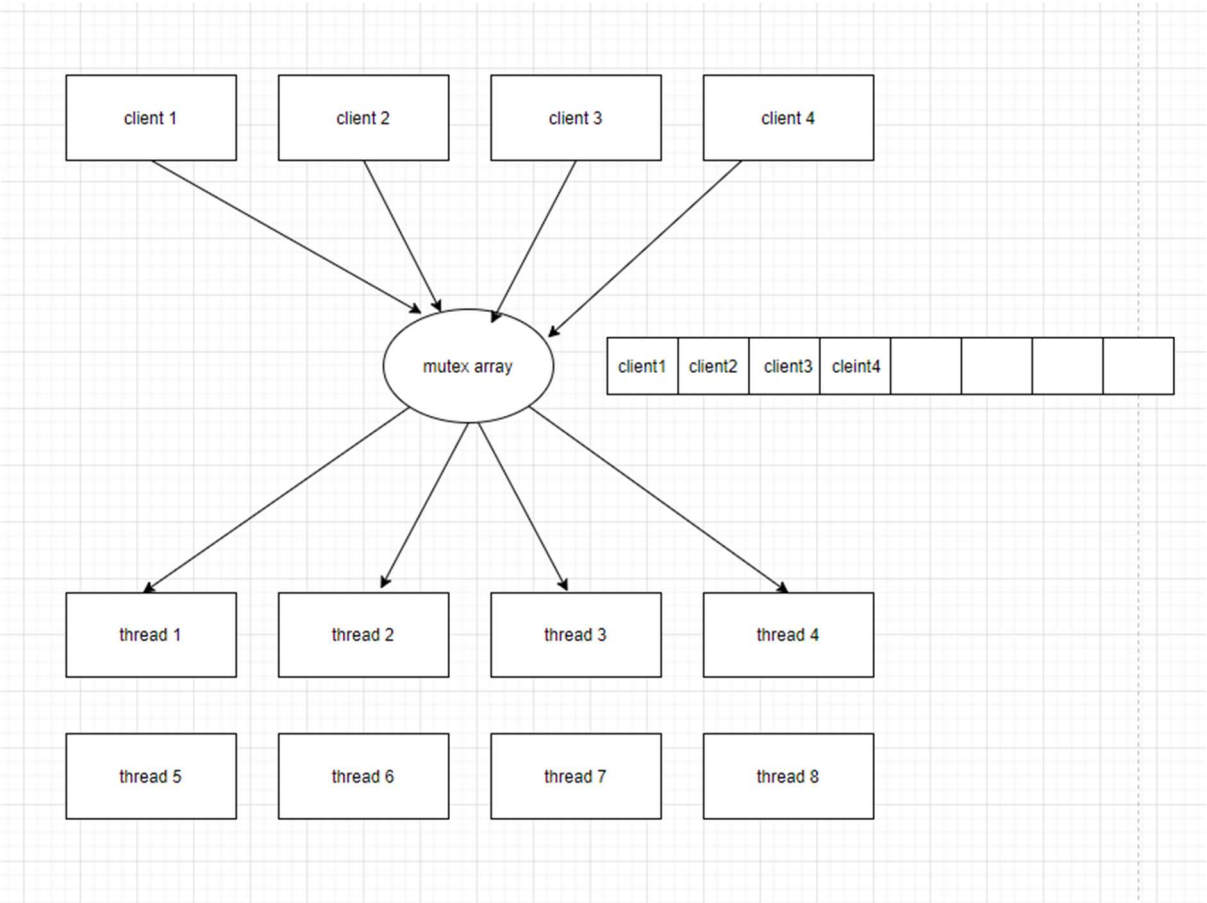
A. Flow Chart

- **2.B.개발 내용에 대한 Flow Chart를 작성.**
- **(각각의 방법들(select, phtread)의 특성이 잘 드러나게 그리면 됨.)**

1. select



2. pthread



B. 제작 내용

- **II. B. 개발 내용의 실질적인 구현에 대해 코드 관점에서 작성.**
- **개발상 발생한 문제나 이슈가 있으면 이를 간략히 설명하고 해결책에 대해 설명.**

1. **select**

1) 처음으로 온 친구를 듣고 있을 file discripter 하나를 만들어준다. 이 친구는 처음 보는 client에 대해서만 구현이 가능하다.

2) 처음으로 온 친구가 file discripter에 추가되었을 경우, 새로운 filediscripter와 client를 연결해 준다.

3) 그 이후에는 새로운 filediscripter까지 보면서 이벤트가 발생했을 경우 이벤트를 처리해준다.(selelct)

4) num_client가 0이 된 경우 파일에 지금까지 썼던 이진트리를 엽데이트 시켜준다.

2. **pthread**

1) 쓰레드를 만들어준다. 만들어지는 쓰레드의 개수는 12개로 고정되어 있으며, 받을 수 있는 클라이언트의 수도 12개로 고정해두었다.

2) 클라이언트의 수가 받을 수 있는 쓰레드를 넘어서는 경우 mutex lock 이 걸린다.

3) 쓰레드에서 처리를 해줘야하는데, show는 reader의 역할, sell, buy는 writer의 역할을 담당한다. show는 다른 명령어에 비해 우선순위가 높게 가져가진다.

show명령어를 위해 여러줄을 보내야하는 이슈가 있었다. 이 이슈는 client side에 개행문자 하나만 들어올 경우 break, server side에서는 입력이 끝났을 경우 마지막에 개행을 넣어줌으로써 해결하였다.

C. 시험 및 평가 내용

시험 전 예측)

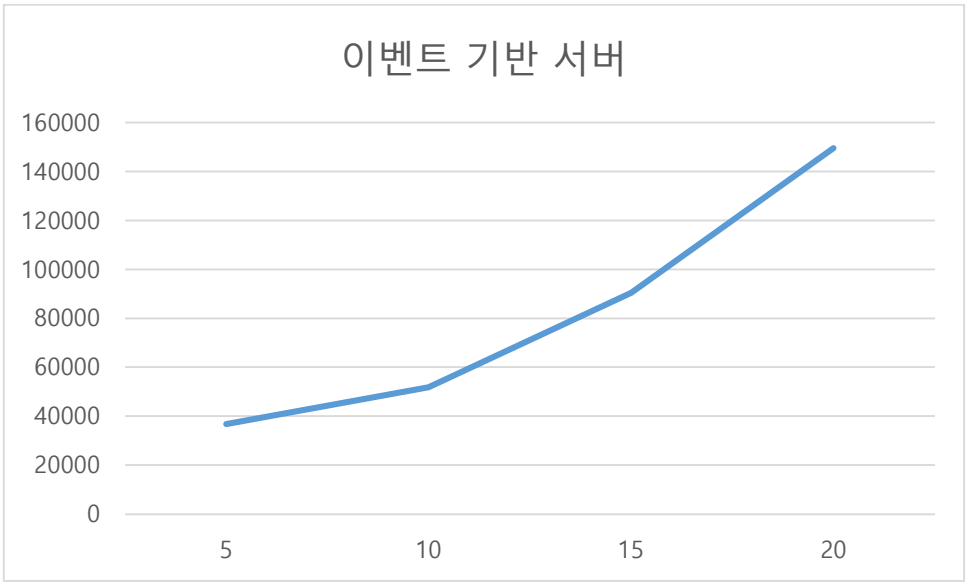
이벤트 기반 서버는 부하가 늘어갈수록 쓰레드보다 느려질 것이다.

부하가 작을 때에는 이벤트 기반과 쓰레드 기반의 수행속도의 차이가 거의 없겠지만,

부하가 늘어난다면, 이벤트 기반은 프로세스가 하나이기 때문에, 다중 작업을 빠르게 수행할 수 없기 때문이다.

또한, 연산 중 가장 로드가 큰 연산은 show일 것이다. show는 IO가 직접적으로 연관이 많이 되기 때문에, 로드가 가장 클 것으로 예상된다.

select

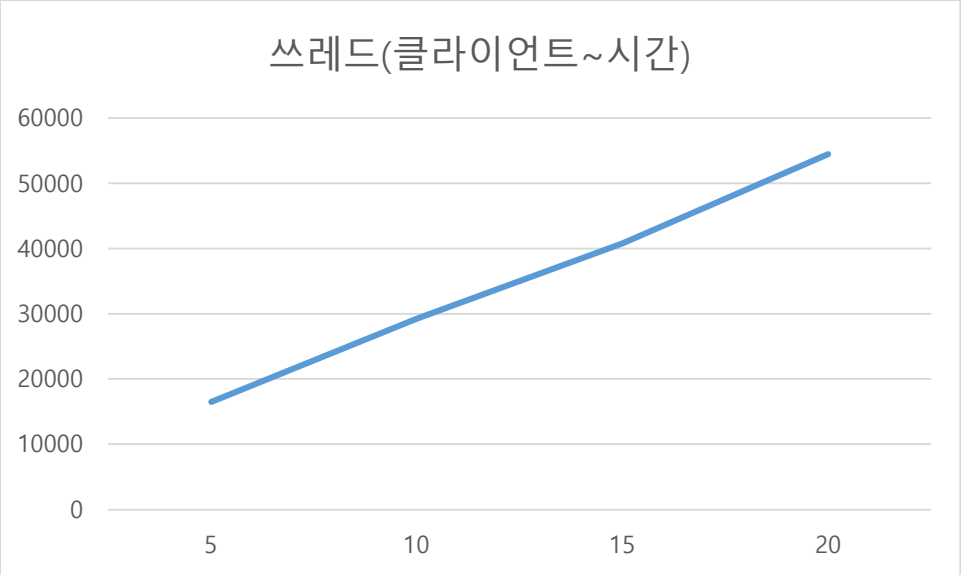


이벤트 기반 서버는 각각의 로드를 각각 이벤트가 발생할때마다 처리를 해주기 때문에,

부하가 큰 작업을 줬을 때 각각의 작업을 하나하나 처리하고 넘어가야하기 때문에, 시간이 다음과 같이 linear한 모양을 띔을 관찰할 수 있다.

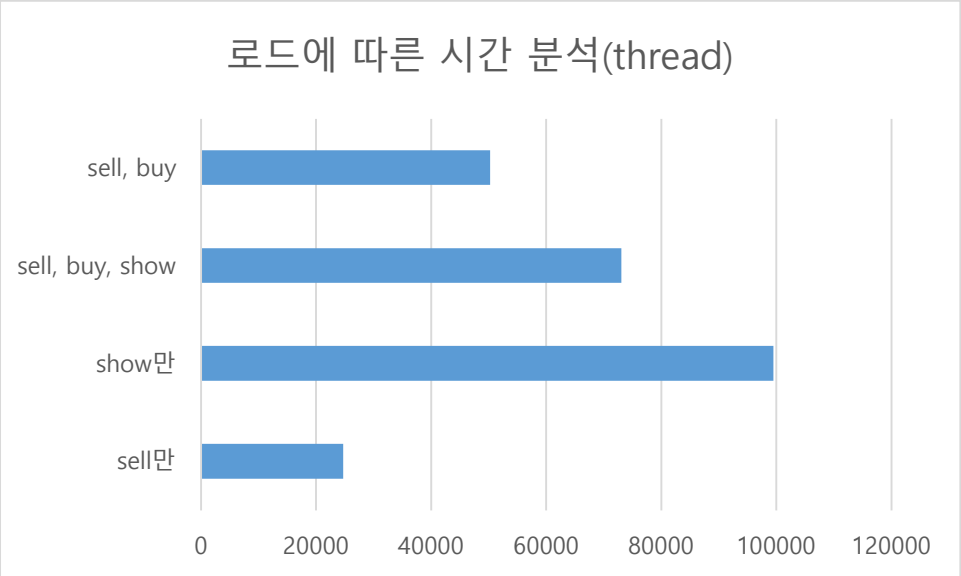
(stock server 500개)

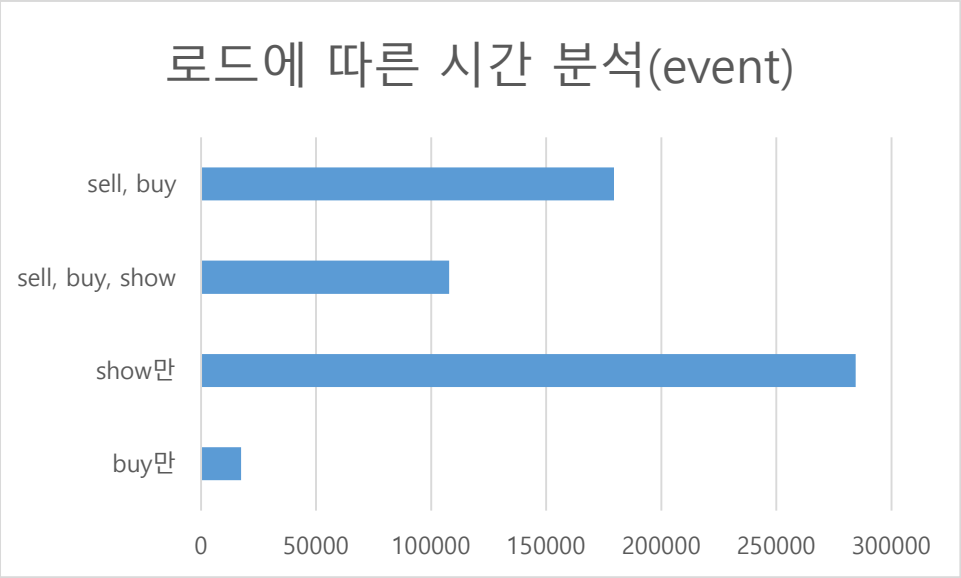
pthread



이 또한 클라이언트가 늘어감에 따라 처리해줘야하는 양이 늘어나서 리니어하게 증가하는 모양을 보인다.

또한, 쓰레드가 더 빠른 모습을 보임을 알 수 있다. 이는 로드를 키울수록 차이는 커질 것으로 보인다.





로드는 입출력과 관련된 show가 가장 많은 시간을 보임을 관찰 할 수 있었고, mutex lock도 크게 시간을 쓰지 않는 모습을 관찰 할 수 있었다.

추가적으로, 쓰레드와 select에서 낮은 워크로드에서는 큰 수행속도의 차이를 보이지 않았다.