System Programming Project 3

담당 교수 : 박성용

이름 : 박민준

학번 : 20212020

1. 개발 목표

- **해당 프로젝트에서 구현할 내용을 간략히 서술.**
- **(주식 서버를 만드는 전체적인 개요에 대해서 작성하면 됨.)**

- 해당 프로젝트의 개발 목표는 효율적으로 주식 정보를 관리하고 클라이언트 요청을 처리하는 주식 서버를 구현하는 것이다. 주식 서버는 클라이언트와의 다중 연결을 지원하며, 각 클라이언트의 요청에 따라 주식 정보를 조회, 구매, 판매할 수 있는 기능을 제공한다. 이를 통해 다수의 클라이언트가 동시에 서버에 접속하여 주식 거래를 원활하게 수행할 수 있다. Event-driven Approach, Thread-based Approach로 구분하여 서버를 구현한다.
- 주요 명령어는 주식 정보 조회(show), 주식 구매(buy), 주식 판매(sell), 연결 종료(exit) 등이 있다. 각 명령어에 대한 응답은 클라이언트에게 전송되며, 클라이언트는 이를 통해 주식 거래를 수행할 수 있다.
- 이 프로젝트를 통해 구현되는 주식 서버는 다중 클라이언트 환경에서의 안정적인 주식 거래 시스템을 제공하며, 성능 최적화 및 동시성 문제를 해결하여 효율적인 데이터 관리를 목표로 한다.

2. 개발 범위 및 내용

A. 개발 범위

- **아래 항목을 구현했을 때의 결과를 간략히 서술**

1. Task 1: Event-driven Approach

- 이벤트 기반 프로그래밍을 사용하여 클라이언트 요청을 비동기적으로 처리하는 주식 서버를 구현한다. select함수를 사용하여 여러 클라이언트의 요청을 동시에 처리할 수 있도록 한다. 비동기 방식으로 요청을 처리하여, 서버는 I/O 작업이 완료될 때까지 기다리지 않고 다른 작업을 수행할 수 있다.
- 이벤트 기반 접근 방식은 I/O 효율성을 높이며, 많은 수의 클라이언트를 동시에 처리할 수 있는 능력을 제공한다. 서버는 블로킹 없이 클라이언트 요청을 빠르게 처리할 수 있어, 높은 동시성(concurrency)을 달성할 수 있다. 그러나 복잡한 상태 관리를 필요로 하며, 코드가 다소 복잡해질 수 있다.

2. Task 2: Thread-based Approach

- 멀티스레드 환경에서 각 클라이언트 요청을 독립적으로 처리하는 주식 서버를 구현한다. 스레드 풀(thread pool)을 사용하여 여러 스레드를 생성하고, 각 스레드는 클라이언트 요청을 처리한다. 주식 데이터의 동시 접근을 제어하기 위해 mutex와 semaphore를 사용하여 동기화 문제를 해결한다.
- 스레드 기반 접근 방식은 코드의 직관성을 높이고, 각 요청을 독립적인 스레드에서 처리함으로써 처리의 병렬성을 높인다. 또한, 스레드 풀을 사용하여 스레드 생성 비용을 줄이고, 효율적인 리소스 관리를 가능하게 한다. 이는 높은 동시성을 제공하지만, 스레드 간 컨텍스트 스위칭 오버헤드가 발생할 수 있으며, 많은 수의 스레드를 처리할 때 시스템 자원 사용이 증가할 수 있다.

3. Task 3: Performance Evaluation

- 성능 평가를 위해 서버의 처리 성능을 측정하고 분석한다. 각 클라이언트 요청을 처리하는 데 걸리는 시간을 측정하고, 시간당 처리 가능한 요청 수를 계산한다. 다양한 시나리오에서 서버의 성능을 테스트하고, 결과를 분석하여 최적화 포인트를 도출한다.
- 성능 평가를 통해 서버의 처리 능력을 객관적으로 파악할 수 있다. 요청 처리 시간과 시간당 처리 가능한 요청 수를 통해 서버의 효율성을 평가하고, 병목 현상을 식별할 수 있다. 이를 바탕으로 서버 성능을 최적화하고, 향후 개선 사항을 도출할 수 있다.

B. 개발 내용

- **아래 항목의 내용만 서술**
- **(기타 내용은 서술하지 않아도 됨. 코드 복사 붙여 넣기 금지)**
- **Task1 (Event-driven Approach with select())**

✓ Multi-client 요청에 따른 I/O Multiplexing 설명

- select() 함수를 사용하여 여러 클라이언트의 소켓을 동시에 감시하고, 준비된 소켓에 대해 I/O 작업을 수행한다. select() 함수는 소켓 파일 descriptor 집합을 입력으로 받아, 어떤 소켓이 읽기, 쓰기 또는 예외 상황에 준비되었는지 확인한다.

- 이벤트 루프에서 select() 함수는 준비된 소켓이 없을 때 블록(block) 상태로 대기하다가, 하나 이상의 소켓이 준비되면 반환하여 해당 소켓에 대한 I/O 작업을 수행한다. 이를 통해 단일 스레드에서 다수의 클라이언트 요청을 효율적으로 처리할 수 있다.

  ✓ epoll과의 차이점 서술

- epoll은 select와 유사한 기능을 제공하지만, 더 많은 수의 파일 descriptor를 효율적으로 처리할 수 있는 고성능 I/O 멀티플렉싱 기법이다. select는 매 호출마다 파일 descriptor 집합을 다시 생성하고 확인해야 하므로, 많은 수의 파일 descriptor를 처리할 때 성능 저하가 발생한다.

- 반면, epoll은 파일 descriptor를 커널 공간에 저장하고, 이벤트가 발생한 파일 descriptor만을 반환하므로, 성능이 더욱 뛰어나다. 또한 epoll은 리눅스에서만 사용할 수 있으며, 대규모 네트워크 서버와 같이 많은 동시 접속을 처리해야 하는 상황에서 주로 사용된다.

- **Task2 (Thread-based Approach with pthread)**

  ✓ Master Thread의 Connection 관리

- Master Thread는 클라이언트의 연결 요청을 수락하고, 새로 연결된 클라이언트 소켓을 작업 큐에 추가하는 역할을 한다. listening 소켓을 감시하다가 새로운 연결 요청이 들어오면 accept() 함수를 호출하여 클라이언트와의 연결을 설정한다. 설정된 클라이언트 소켓은 작업 큐에 삽입되어, 작업 스레드가 이를 처리할 수 있도록 준비된다.

  ✓ Worker Thread Pool 관리하는 부분에 대해 서술

- 작업 스레드 풀은 고정된 수의 스레드로 구성되며, 각 스레드는 작업 큐에서 클라이언트 요청을 가져와 처리한다. 작업 스레드는 대기 상태에서 작업 큐에 새로운 클라이언트 요청이 추가되면, 이를 꺼내어 I/O 작업을 수행한다.

- 작업 큐는 생산자-소비자 모델을 사용하여, Master Thread가 클라이언트 소켓을 큐에 추가하고, 작업 스레드가 이를 처리하는 구조로 동작한다. 또한, mutex와 semaphore를 사용하여 작업 큐의 동기화 문제를 해결하고, 스레드 간의 안전한 데이터 접근을 보장한다.

- **Task3 (Performance Evaluation)**

  ✓ 얻고자 하는 metric 정의, 그렇게 정한 이유, 측정 방법 서술

  [metric 정의]

- 소요 시간 (Elapsed Time): 클라이언트 요청을 처리하는 데 걸리는 총 시간. 시스템의 응답성을 평가할 수 있다.
- 초당 요청 수 (Requests per second): 단위 시간당 처리할 수 있는 클라이언트 요청의 수. 시스템의 처리 능력을 평가할 수 있다.

  [이유]

- Elapsed Time은 서버의 응답 속도를 직접적으로 나타내며, 클라이언트가 요청을 보내고 응답을 받는 데 걸리는 시간을 측정한다.
- Requests per second는 서버의 처리 성능을 나타내며, 단위 시간당 얼마나 많은 요청을 처리할 수 있는지를 평가하는 중요한 지표이다.

  [측정 방법]

- 서버 시작 시 타이머를 시작하고, 클라이언트 요청을 처리하는 동안 요청 수를 카운트한다. 클라이언트 요청 처리가 완료된 후 타이머를 중지하고, 총 소요 시간을 계산한다. 총 요청 수를 소요 시간으로 나누어 초당 요청 수를 계산한다.

  ✓ Configuration 변화에 따른 예상 결과 서술

- 클라이언트 수가 증가하면, 초기에는 초당 요청 수가 증가할 것으로 예상된다. 이는 멀티스레드 환경에서 병렬 처리가 효과적으로 이루어지기 때문이다. 그러나 일정 수준을 넘어서면 초당 요청 수가 감소할 것이다. 이는 클라이언트 수가 급격히 증가함에 따라 스레드 간의 context switching overhead와 동기화 비용이 증가하고, 서버가 과부하 상태에 도달하여 각 요청을 처리하는 데 더 오랜 시간이 걸리기 때문이다.

C. 개발 방법

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

✓ **Task1 (Event-driven Approach with select())**

- pool 구조체를 정의하여 클라이언트 파일 descriptor와 I/O 버퍼를 관리한다. pool 구조체는 최대 파일 descriptor, 파일 descriptor 집합, 준비된 파일 descriptor 수, 클라이언트 파일 descriptor 배열 및 I/O 버퍼 배열을 포함한다.
- 주식 정보를 저장하는 item 구조체와 이진 트리 TreeNode 구조체를 정의한다.
- init_pool(), add_client(), check_clients() 함수들을 추가하여 클라이언트 연결을 관리하고, 클라이언트 요청을 처리한다. init_pool()은 초기화 함수로, pool 구조체를 초기화하고 listening파일 descriptor를 설정한다. add_client() 함수는 새로운 클라이언트 연결을 추가하고, 파일 descriptor 집합에 추가한다. check_clients() 함수는 준비된 클라이언트 요청을 처리하고, 요청에 따라 주식 정보를 조회, 구매, 판매한다.
- sigint_handler() 함수를 추가해 서버 종료 시 주식 데이터를 저장하고 안전하게 종료하기 위한 시그널 핸들러를 설정한다.
- main 함수에서는 서버를 초기화하고, 클라이언트 요청을 처리한다. 여기서 클라이언트 풀을 초기화하고, select를 사용하여 이벤트를 감시한다.
- load_stock_data() 함수를 추가해 주식 데이터를 로드, 저장하고 클라이언트의 명령에 따라 주식 정보를 처리하는 함수를 구현한다. 해당 함수에서는 파일에서 주식 데이터를 로드하여 이진 트리에 삽입하는 과정을 처리한다.
- save_stock_data() 함수를 추가해 현재 주식 데이터를 파일에 저장한다. 또한, save_stock_data_recursive() 함수를 이용해 재귀적으로 트리의 모든 노드를 순회하며 데이터를 저장하는 과정을 처리한다.
- 주식 정보를 이진 트리에 삽입하고, 검색하기 위해 다음 함수를 정의한다.

  TreeNode* insert_stock(TreeNode *node, item stock);

  TreeNode* find_stock(TreeNode *node, int ID);

- show_stock() 함수 및 show_stock_recursive() 함수를 추가해 주식 정보를 클라이언트에 출력하고, 트리를 순회하며 모든 주식 정보를 버퍼에 저장한다. 트리를 중위 순회(in-order traversal)하여 주식 정보를 버퍼에 저장하는 방식이다.
- 마지막으로 buy_stock() 함수, sell_stock() 함수는 주식을 구매하고 판매하는 함수

이다. buy_stock() 함수는 클라이언트의 주식 구매 요청을 처리하는데, 주어진 ID에 해당하는 주식을 찾아서 수량을 감소시킨다. 만약 수량이 충분하지 않으면 오류 메시지를 반환한다. 이와 반대로 sell_stock() 함수는 클라이언트의 주식 판매 요청을 처리하는데, 주어진 ID에 해당하는 주식을 찾아서 수량을 증가시킨다.

✓ **Task2 (Thread-based Approach with pthread)**

- 가장 먼저, Task1과 마찬가지로 주식 정보를 저장하는 item 구조체를 정의하고, 주식 정보를 저장하는 트리의 루트와 트리 전체에 대한 접근을 제어하는 semaphore를 포함하는 stock_tree_t 구조체를 정의한다.
- 이후 작업 큐를 관리하기 위해 sbuf_t 구조체를 정의한다. sbuf_t 구조체는 연결된 파일 descriptor를 저장하는 공유 버퍼이다. 버퍼는 semaphore를 사용하여 스레드 간의 동기화를 제공한다. 이후 sbuf_init(), sbuf_insert(), sbuf_remove(), thread() 함수들을 추가하여 작업 큐를 초기화하고 관리한다.
- sbuf_init()은 작업 큐를 초기화하는 함수로, 큐의 크기를 설정하고 필요한 semaphore를 초기화한다. sbuf_insert()는 새로운 클라이언트 소켓을 큐에 삽입하는 함수로, 슬롯을 확보하고 큐에 추가한다. sbuf_remove()는 작업 큐에서 클라이언트 소켓을 제거하는 함수로, 아이템을 제거하고 반환한다. thread()는 작업 스레드 함수로, 큐에서 클라이언트 소켓을 가져와 요청을 처리한다.
- insert_item() 함수를 추가해 주식 정보를 이진 트리에 삽입한다. 또한, find_item() 함수는 주어진 ID에 해당하는 주식 정보를 트리에서 찾아주는 함수이다.
- load_items() 함수와 save_items() 함수는 서로 반대의 역할을 하는 함수로, load_items()는 파일에서 주식 데이터를 로드하여 이진 트리에 삽입하고 save_items()는 현재 주식 데이터를 파일에 저장한다. 이 때, 재귀적으로 트리의 모든 노드를 순회하며 데이터를 저장한다.
- sigint_handler() 함수를 설정해 서버 종료 시 주식 데이터를 저장하고 안전하게 종료한다.
- handle_client() 함수를 추가해 show, buy, sell 등 클라이언트의 요청을 처리한다. while문을 돌면서 각각의 요청 결과를 큰 버퍼에 한꺼번에 저장해 두었다가, while문이 끝나는 시점에 한 번에 결과를 출력하는 방식을 이용한다.
- 마지막으로 preorder() 함수는 트리를 전위 순회하며 주식 정보를 버퍼에 저장하

는 함수이다.

✓ **Task3 (Performance Evaluation)**

- 기존의 multiclient.c 파일을 수정하여 서버의 성능을 평가하고 분석한다. 먼저, main() 함수 첫 부분에 성능 측정을 위해 시작 시간과 종료 시간을 기록하기 위한 시간 변수(struct timeval start, end; double elapsed_time;)를 추가한다.
- 클라이언트 프로세스를 생성하는 while문 전에 gettimeofday(&start, NULL); 함수를 호출하여 시작 시간을 기록한다.
- 모든 클라이언트 프로세스가 종료된 후 main() 함수가 종료 되기 전에, 종료 시간(gettimeofday(&end, NULL);)을 기록하고, 소요 시간과 초당 요청 수를 계산하여 출력한다. 소요 시간은 다음과 같이 계산한다.

```
elapsed_time = (end.tv_sec - start.tv_sec) + ((end.tv_usec - start.tv_usec) / 1000000.0);
```

3. 구현 결과

- **2번의 구현 결과를 간략하게 작성**
- **미처 구현하지 못한 부분에 대해선 디자인에 대한 내용도 추가**

✓ **Task1 (Event-driven Approach with select())**

- select() 함수를 사용하여 다수의 클라이언트 요청을 동시에 처리하는 서버를 구현하였다. 클라이언트의 연결 요청을 감지하고, 각 클라이언트의 요청을 처리할 수 있는 이벤트 기반 서버가 성공적으로 동작하였다. 각 클라이언트는 show, buy, sell 명령을 서버에 보내고, 서버는 이를 처리하여 응답을 반환한다.
- Task1에서 epoll을 사용한 구현은 포함되지 않았다. epoll을 사용하면 대규모 네트워크 서버에서 더 효율적인 I/O 멀티플렉싱을 제공할 수 있다. epoll의 디자인에서는 이벤트를 등록하고 대기하는 방식으로, select()에 비해 더 많은 클라이언트를 효율적으로 처리할 수 있다는 장점이 존재한다.

✓ **Task2 (Thread-based Approach with pthread)**

- 멀티스레드 기반의 서버를 구현하여 각 클라이언트 요청을 처리하는 작업 스레드

풀을 만들었다. Master 스레드는 클라이언트의 연결 요청을 받아서 작업 큐에 추가하고, 작업 스레드는 큐에서 클라이언트 요청을 가져와 처리한다. 각 클라이언트는 show, buy, sell 명령을 서버에 보내고, 서버는 이를 처리하여 응답을 반환한다.

- Task2에서 동적 스레드 풀 관리 기능은 구현되지 않았다. 동적 스레드 풀은 요청의 수에 따라 스레드의 수를 조절하여 자원 사용을 최적화할 수 있다. 동적 스레드 풀의 디자인에서는 요청이 많아지면 스레드를 추가하고, 요청이 적어지면 스레드를 줄이는 방식으로 동작한다.

✓ **Task3 (Performance Evaluation)**

- 여러 클라이언트 프로세스를 생성하여 서버의 성능을 평가했다. 각 클라이언트는 show, buy, sell 명령을 랜덤하게 서버에 보내고, 서버의 응답을 받아 처리하는 과정을 반복한다. 성능 평가 결과, 총 소요 시간과 초당 요청 처리량을 측정하여 출력하였다.
- Task3에서는 네트워크 지연 시간 및 서버 부하를 고려한 더 다양한 성능 지표를 측정하지 못했다. 추가로 고려할 수 있는 성능 지표로는 평균 응답 시간, 최대 응답 시간, 각 명령어별 처리 시간 등이 있다. 이러한 성능 지표를 측정하기 위한 디자인에서는 클라이언트와 서버 간의 네트워크 트래픽을 모니터링하고, 각 요청과 응답의 시간을 기록하여 분석하는 방식을 사용할 수 있다.

4. 성능 평가 결과 (Task 3)

- **강의자료 슬라이드의 내용 참고하여 작성 (측정 시점, 출력 결과 값 캡처 포함)**

[실험 환경]

- server: cspro, client: cspro9, port number: 60065
- multiclient.c 파일에서 MAX_CLIENT를 1000으로, ORDER_PER_CLIENT는 100으로 설정하였다.
- 성능 평가 및 분석 결과를 빠르게 얻기 위해 multiclient.c 파일에서 Fputs() 함수와 usleep() 함수를 주석 처리하였다.

✓ **확장성: 각 방법에 대한 Client 개수 변화에 따른 동시 처리율 변화 분석**

- Client의 개수를 1, 10, 50, 100, 500, 1000 총 6가지 경우의 수로 나누어 소요 시간 및 초당 client 처리 요청 개수를 측정하였다.

1) Event-driven Approach

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 1
Elapsed time            : 0.041031 s
Requests per second     : 2437.181643
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 10
Elapsed time            : 0.083267 s
Requests per second     : 12009.559609
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 50
Elapsed time            : 0.378892 s
Requests per second     : 13196.372581
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 100
Elapsed time            : 0.724616 s
Requests per second     : 13800.412908
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.521651 s
Requests per second     : 14197.886162
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 1000
Elapsed time            : 7.034821 s
Requests per second     : 14215.002770
```

- 측정 결과 Client의 개수가 증가함에 따라 소요 시간 및 초당 client 처리 요청 개수 또한 함께 비례하여 증가한 것을 알 수 있다.

2) Thread-based Approach

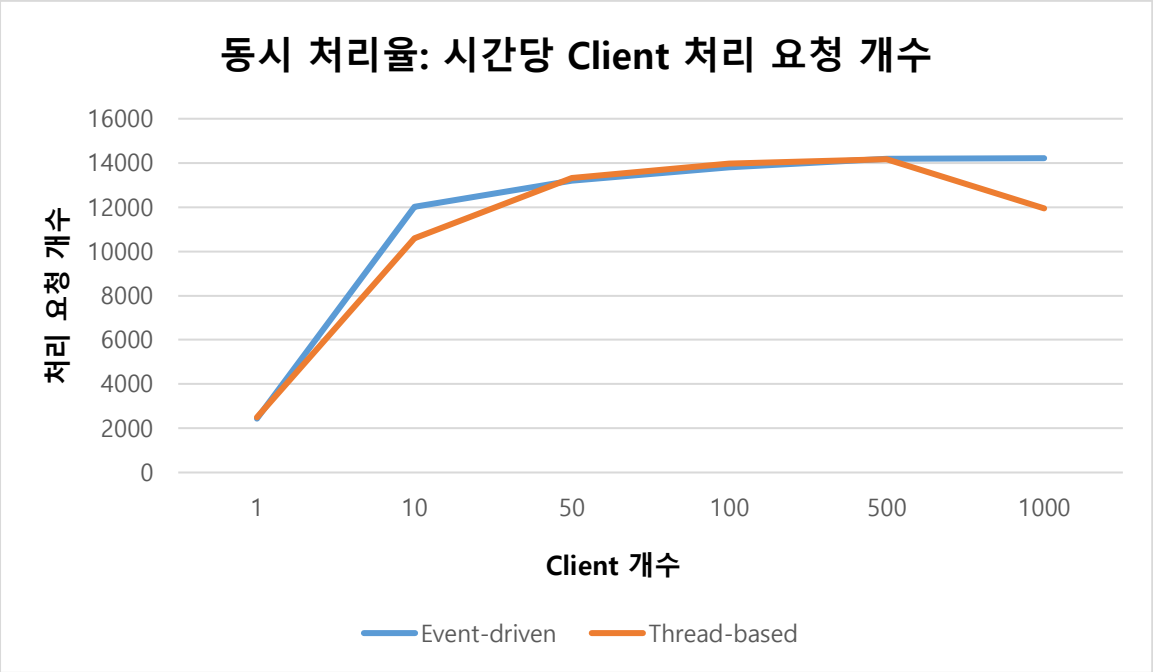
- NTHREADS는 8, SBUFSIZE는 64로 설정하였다.

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 1
Elapsed time            : 0.040070 s
Requests per second     : 2495.632643
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 10
Elapsed time            : 0.094491 s
Requests per second     : 10583.018489
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 50
Elapsed time            : 0.375279 s
Requests per second     : 13323.420708
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 100
Elapsed time            : 0.715397 s
Requests per second     : 13978.252635
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.528942 s
Requests per second     : 14168.552501
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 1000
Elapsed time            : 8.368318 s
Requests per second     : 11949.832690
```

- 측정 결과, Even-driven Appraoch의 결과와 마찬가지로 Client의 개수가 증가함에 따라 소요 시간 및 초당 client 처리 요청 개수가 비례하여 증가하였다.

- 해당 결과를 그래프로 나타내 비교해보면 다음과 같다.

(처리 요청 개수는 소수점 첫째 자리에서 반올림하였다.)



- Client 개수가 증가함에 따라 처음에는 Event-driven server의 동시 처리율이 Thread-based server보다 높았지만 Client 개수가 약 50개 이상이 되는 순간 Thread-based server의 동시 처리율이 더 높아졌다. 이후 두 server 모두 동시 처리율이 미묘하게 증가하지만, Client 개수가 약 500개 이상이 되자 Event-driven server의 동시 처리율은 계속해서 증가하는 반면 Thread-based server는 오히려 감소하는 것을 확인할 수 있었다.

- 이러한 결과가 나타나는 이유는 각 서버의 특성 및 장단점과 관련이 있다.

- client 개수가 약 50개 이하인 경우 Event-driven server가 단일 스레드로 효율적으로 동작하며, 낮은 오버헤드로 인해 높은 동시 처리율을 보인다. 반면, Thread-based server는 스레드 생성 및 context switching 오버헤드로 인해 Event-driven server보다 동시 처리율이 낮다.

- client 개수가 약 50개 이상인 경우 Event-driven server는 client 수가 증가하면서 단일 스레드의 한계에 도달하게 된다. 특히, 특정 클라이언트의 요청이 오래 걸리면 다른 요청들이 지연될 수 있다. 반면, Thread-based server는 다수의 워커 스레드를 통해 병렬 처리를 하기 때문에 클라이언트 요청을 보다 효율적으로 처리할 수 있다. 이로 인해 동시

처리율이 Event-driven server를 넘어선다.

- 마지막으로 client 개수가 약 500개 이상인 경우 Thread-based server의 오버헤드가 극대화된다. 스레드 수가 많아지면서 스레드 관리와 context switching으로 인한 오버헤드가 급격히 증가하여 동시 처리율이 감소한다. 또한, 메모리 사용량도 급격히 증가하여 시스템 리소스가 한계에 도달할 수 있다. 반면, Event-driven server는 여전히 단일 스레드로 동작하지만, epoll과 같은 효율적인 이벤트 감지 메커니즘 덕분에 많은 클라이언트의 연결을 상대적으로 효율적으로 관리할 수 있다. 이로 인해 client 수가 많아지더라도 동시 처리율이 계속 증가하는 모습을 보인다.

✓ **워크로드에 따른 분석: Client 요청 타입 (buy, show, sell 등)에 따른 동시 처리율 변화 분석**

- Client의 개수를 500으로 고정하여 소요 시간 및 초당 client 처리 요청 개수를 측정하였다.
- 각 워크로드마다 10번씩 반복하여 multiclient 실행하였고, 이를 통해 평균값을 계산하여 실행 결과의 신뢰성을 높였다.

1) Event-driven Approach

(1) 모든 client가 show만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.554999 s
Requests per second     : 14064.701565
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.548388 s
Requests per second     : 14090.905504
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.541565 s
Requests per second     : 14118.052330
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.548284 s
Requests per second     : 14091.318508
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545028 s
Requests per second     : 14104.260954
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545151 s
Requests per second     : 14103.771602
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.540500 s
Requests per second     : 14122.299110
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545188 s
Requests per second     : 14103.624406
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.544789 s
Requests per second     : 14105.211904
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.549836 s
Requests per second     : 14085.157737
```

(2) 모든 client가 buy만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.539950 s
Requests per second     : 14124.493284
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.560325 s
Requests per second     : 14043.661744
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.552431 s
Requests per second     : 14074.868731
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.544376 s
Requests per second     : 14106.855480
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.544756 s
Requests per second     : 14105.343217
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.542665 s
Requests per second     : 14113.668665
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.546374 s
Requests per second     : 14098.907786
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.548527 s
Requests per second     : 14090.353547
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.542699 s
Requests per second     : 14113.533213
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.538081 s
Requests per second     : 14131.954582
```

(3) 모든 client가 sell만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.793236 s
Requests per second     : 13181.357553
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.921388 s
Requests per second     : 12750.587292
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 7.925120 s
Requests per second     : 6309.052734
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 4.178402 s
Requests per second     : 11966.297163
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 4.752572 s
Requests per second     : 10520.619151
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.662559 s
Requests per second     : 13651.657216
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.529756 s
Requests per second     : 14165.285079
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.534842 s
Requests per second     : 14144.903789
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 4.075320 s
Requests per second     : 12268.975197
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.711155 s
Requests per second     : 13472.894557
```

(4) 모든 client가 buy 또는 show를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.578210 s
Requests per second     : 13973.467181
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.533826 s
Requests per second     : 14148.970549
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.543399 s
Requests per second     : 14110.745078
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.526601 s
Requests per second     : 14177.957756
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.671901 s
Requests per second     : 13616.924857
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.562978 s
Requests per second     : 14033.204808
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.534819 s
Requests per second     : 14144.995826
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.549230 s
Requests per second     : 14087.562654
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.603101 s
Requests per second     : 13876.935451
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.556729 s
Requests per second     : 14057.860467
```

(5) 모든 client가 sell 또는 show를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.527108 s
Requests per second     : 14175.919762
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.595400 s
Requests per second     : 13906.658508
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.745346 s
Requests per second     : 13349.901451
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.540561 s
Requests per second     : 14122.055799
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.531297 s
Requests per second     : 14159.103581
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.543798 s
Requests per second     : 14109.156335
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.754780 s
Requests per second     : 13316.359414
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.524665 s
Requests per second     : 14185.745312
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.546222 s
Requests per second     : 14099.512100
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.553125 s
Requests per second     : 14072.119613
```

(6) 모든 client가 buy 또는 sell를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.540054 s
Requests per second     : 14124.078333
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.577529 s
Requests per second     : 13976.127098
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.529630 s
Requests per second     : 14165.790749
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.525089 s
Requests per second     : 14184.039041
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.539744 s
Requests per second     : 14125.315277
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.536708 s
Requests per second     : 14137.440807
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.606742 s
Requests per second     : 13862.926708
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.751429 s
Requests per second     : 13328.254380
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.744360 s
Requests per second     : 13353.416872
cse20212020@cspro9:~/SP/Lab3/task_1$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.533355 s
Requests per second     : 14150.856622
```

- 대부분의 경우 10번 반복 시 비슷한 결과가 나왔지만, 모든 client가 sell만 요청하는 경우 7초 또는 4초 등 10번 중 1~2번 정도 더 높은 값이 출력되었다.

- 각 경우에 대하여 평균값을 계산하고, 이를 표로 나타내면 다음과 같다.

(평균값은 소수점 첫째 자리에서 반올림하였다.)

워크로드 예제	동시 처리율 (단위: 개)
show only	14,099
buy only	14,100
sell only	12,243
buy or show	14,023
sell or show	13,950
buy or sell	13,941

2) Thread-based Approach

(1) 모든 client가 show만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 5.221875 s
Requests per second     : 9575.104728
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.508807 s
Requests per second     : 14249.857573
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.520809 s
Requests per second     : 14201.281580
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.572767 s
Requests per second     : 13994.755325
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.530367 s
Requests per second     : 14162.833496
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.529377 s
Requests per second     : 14166.806210
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.513412 s
Requests per second     : 14231.180402
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.517248 s
Requests per second     : 14215.659516
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.519104 s
Requests per second     : 14208.162078
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.511628 s
Requests per second     : 14238.410219
```

(2) 모든 client가 buy만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545704 s
Requests per second     : 14101.571930
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.544094 s
Requests per second     : 14107.977949
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.569449 s
Requests per second     : 14007.764224
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545700 s
Requests per second     : 14101.587839
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.562861 s
Requests per second     : 14033.665641
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.544124 s
Requests per second     : 14107.858529
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.564091 s
Requests per second     : 14028.822496
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.551879 s
Requests per second     : 14077.056116
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.552209 s
Requests per second     : 14075.748358
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.554898 s
Requests per second     : 14065.101165
```

(3) 모든 client가 sell만 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 6.116359 s
Requests per second     : 8174.798111
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 5.662902 s
Requests per second     : 8829.395246
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 7.217270 s
Requests per second     : 6927.827281
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 7.711130 s
Requests per second     : 6484.133973
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 7.887574 s
Requests per second     : 6339.084743
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 5.685512 s
Requests per second     : 8794.282731
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.927910 s
Requests per second     : 12729.415898
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.545374 s
Requests per second     : 14102.884491
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.517755 s
Requests per second     : 14213.610669
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.619587 s
Requests per second     : 13813.730683
```

(4) 모든 client가 buy 또는 show를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.588875 s
Requests per second     : 13931.942461
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.534920 s
Requests per second     : 14144.591674
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.514385 s
Requests per second     : 14227.240328
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.522849 s
Requests per second     : 14193.057948
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.515734 s
Requests per second     : 14221.781284
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.511991 s
Requests per second     : 14236.938534
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.512063 s
Requests per second     : 14236.646666
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.512574 s
Requests per second     : 14234.575556
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.879958 s
Requests per second     : 12886.737434
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.518381 s
Requests per second     : 14211.081745
```

(5) 모든 client가 sell 또는 show를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.510351 s
Requests per second     : 14243.589886
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.507154 s
Requests per second     : 14256.573849
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.514804 s
Requests per second     : 14225.544298
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.509958 s
Requests per second     : 14245.184700
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.507140 s
Requests per second     : 14256.630759
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.506951 s
Requests per second     : 14257.399091
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.513538 s
Requests per second     : 14230.670054
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.512195 s
Requests per second     : 14236.111605
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.509715 s
Requests per second     : 14246.170985
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.561095 s
Requests per second     : 14040.625145
```

(6) 모든 client가 buy 또는 sell를 요청하는 경우

```
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.523608 s
Requests per second     : 14190.000704
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.577656 s
Requests per second     : 13975.630972
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.521046 s
Requests per second     : 14200.325699
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.519466 s
Requests per second     : 14206.700676
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.525521 s
Requests per second     : 14182.300999
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.519179 s
Requests per second     : 14207.859276
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.525131 s
Requests per second     : 14183.870046
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.512925 s
Requests per second     : 14233.153284
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.512877 s
Requests per second     : 14233.347766
cse20212020@cspro9:~/SP/Lab3/task_2$ ./multiclient 172.30.10.11 60065 500
Elapsed time            : 3.511710 s
Requests per second     : 14238.077746
```

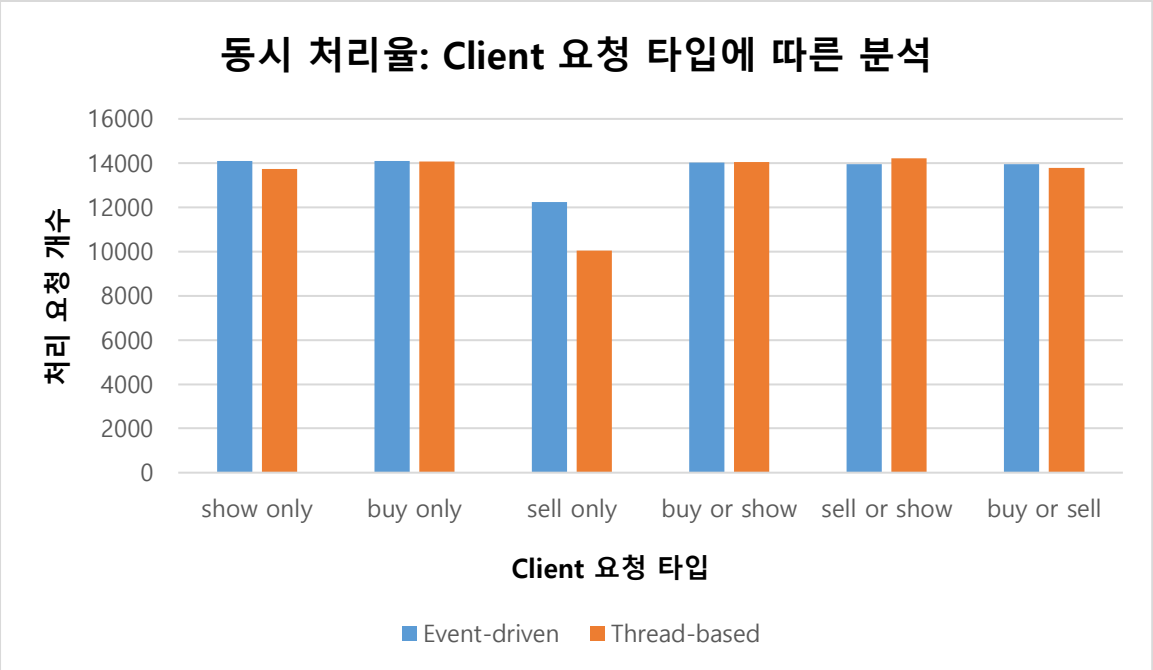
- Thread-based Approach도 Event-driven Approach와 마찬가지로 10번 반복 시 비슷한 결과가 나왔지만, 모든 client가 sell만 요청하는 경우 값의 변동이 가장 컸다.

- 각 경우에 대하여 평균값을 계산하고, 이를 표로 나타내면 다음과 같다.

(평균값은 소수점 첫째 자리에서 반올림하였다.)

워크로드 예제	동시 처리율 (단위: 개)
show only	13,724
buy only	14,071
sell only	10,041
buy or show	14,052
sell or show	14,224
buy or sell	13,785

- Event-driven Approach와 Thread-based Approach 두 가지 경우에 대한 워크로드 예제 결과 값을 비교해보면 다음과 같은 그래프를 얻을 수 있다.



- 분석 결과, 대부분의 경우 두 server 모두 비슷한 값을 갖지만 모든 client과 sell만 요청하는 경우에는 Thread-based server가 Event-driven server보다 더 낮은 동시 처리율을 갖는다.

- 이러한 분석 결과가 나타나는 이유는 Thread-based server의 데이터 수정 및 동기화와 관련이 있다. sell 요청을 처리하기 위해서는 server의 주식 데이터를 수정해야 한다. 이 과정에서 동시 접근으로 인한 데이터 일관성을 유지하기 위해 mutex나 semaphore를 사용한다. 즉, Thread-based server에서는 여러 스레드가 동시에 동일한 주식 데이터를 수정하려고 할 때, mutex나 semaphore로 인해 스레드들이 대기 상태에 들어간다. 이로 인해 스레드 간의 context switching과 잠금/해제 오버헤드가 증가하여 처리율이 낮아지게 된다.

- show 요청은 단순히 데이터를 읽는 작업이기 때문에, 데이터 수정과 동기화가 필요하지 않다. 따라서 Thread-based server와 Event-driven server 모두 높은 동시 처리율을 보인다. buy 요청도 데이터 수정이 필요하지만, sell 요청에 비해 상대적으로 lock 경합이 덜 발생할 수 있다. 이는 sell 요청이 데이터 수정 빈도가 더 높고, client 요청이 많을 때 lock 경합이 더 자주 발생하기 때문이다. buy or show, sell or show, buy or sell 조합에서도 비슷한 이유로 동시 처리율이 비교적 높게 나타난다.

- 결론적으로, 모든 클라이언트가 sell만 요청하는 경우 Thread-based server의 데이터 수정과 동기화 비용이 증가하여 스레드 간의 대기 시간이 길어지고, 여러 스레드가 동일한 자원을 수정하려고 할 때 발생하는 lock 경합으로 인해 성능이 저하되어 동시 처리율이 Event-driven 서버보다 낮아진다. 이와 달리, Event-driven server는 단일 스레드로 모든 요청을 처리하기 때문에 이러한 오버헤드가 적어, client 수가 많아질수록 상대적으로 더 높은 동시 처리율을 유지할 수 있다.