



IOCP - 2

MM4220 게임서버 프로그래밍
정내훈

지금 까지

- 싱글스레드 IOCP 서버 제작
- 멀티스레드 프로그래밍 학습
- 멀티쓰레드 IOCP

IOCP

● 시작

- GetQueuedCompletionStatus를 별도의 쓰레드로 옮기기
- 해보자!
- worker thread 작성
 - 개수? : 코어의 개수
 - `this_thread::hardware_concurrency()`
- Main함수의 GQCS루프를 worker thread로 이전

IOCP

- 문제점

- Thread는 별도의 함수 이므로 많은 변수를 전역변수로 이전해야 한다.

```
array < CLIENT, MAX_USER> clients;  
// 또는 unordered_map<int, CLIENT> clients  
SOCKET g_s_socket;  
HANDLE g_h_iocp;
```

DATA RACE

- Data Race?

- 모든 전역 변수

```
array < CLIENT, MAX_USER> clients;  
SOCKET g_s_socket;  
HANDLE g_h_iocp;
```

- clients : Data Race!
 - 서로 다른 스레드에서 broadcasting 시 접근
- h_iocp, g_s_socket : No Data Race!
 - 멀티스레드 시작 전 초기화된 후 변경되지 않음
 - 지역변수 유지, worker thread에 parameter로 전달

DATA RACE

```
array <CLIENT, MAX_USER> clients;
```

- 엄청난 Data Race

- 거의 모든 함수에서 clients를 건드림
- Clients 컨테이너 자체는 수정되지 않기 때문에 Data Race가 없음
 - 컨테이너의 크기가 변하지 않고, 원소들의 메모리 주소가 변하지 않음. (vector와 map이 곤란한 이유)
- 원소의 값에 접근할 때는 여전히 Data Race존재 => 원소마다 mutex 필요
- 원소의 재사용 문제가 발생, id 재사용 문제도 같이 발생

DATA RACE

```
unordered_map <int, CLIENT> clients;
```

- 엄청난 Data Race
 - 거의 모든 함수에서 clients를 건드림
 - **Clients 컨테이너 자체가 수정**되므로 Data Race대량 발생
 - 원소들의 메모리 주소는 바뀌지 않음. 하지만 erase 시 재사용.
 - 원소 내부에 접근할 때도 여전히 Data Race존재 => 원소마다 mutex 필요
- 해결 방법 #1

```
concurrent_unordered_map <int, CLIENT> clients;
```

- Erase가 안된다

- 해결 방법 #2

```
concurrent_unordered_map <int,  
    atomic_shared_ptr <CLIENT>> clients;
```

- Container의 크기 한계 문제 => 주기적으로 서버 재시작.

DATA RACE

● SESSION

- m_id : no data race
- m_recv_over : no data race
- m_prev_recv : no data_race
- m_s : no data_race
=> data_race
- m_inuse : data Race
- m_name : data race
- m_x, m_y : data race

```
class CLIENT
{
    int            m_id;
    EXP_OVER      m_recv_over;
    unsigned char m_prev_recv;
    SOCKET        m_s;

    bool          m_inuse;
    char          m_name[MAX_NAME];
    short         m_x, m_y;
};
```

- no data race : 한 순간에 오직 한 스레드에서만 접근 할 때 == 두 개 이상의 스레드에서 동시에 접근하지 않을 때
- no data race : 값이 정해진 후 다른 아무도 값을 변경하지 않을 때

DATA RACE

- 정책

- mutex로 보호한다.
- atomic operation을 사용한다.
 - 뒤에서 다시 다룸
- m_x, m_y는 보호하지 않는다. (성능을 위해 오류를 감수한다.)
- m_name의 접근을 m_inuse 이후로 제한한다.
 - array의 단점 : array가 아니면 생성하기 이전에는 접근하는 것이 불가능 하다.

DATA RACE

● 재사용 문제

- m_socket이 DATA RACE가 된다.
 - clients재사용시
- 해결책
 - 값이 valid한지를 나타내는 변수가 필요하다.
 - m_state 변수 추가 필요. (m_inuse 통합)
 - m_state의 의미
 - ST_FREE : 미사용
 - ST_ALLOC : ACCEPT됨, 아직 cs_login 패킷을 받지 못함, 게임 컨텐츠의 값들이 의미가 없음. 이 클라이언트는 broadcasting에서 제외되어야 함.
 - ST_INGAME : 게임 참여 중

DATA RACE

- 실습 : 지난 주 제작한 IOCP 서버를 수정

DATA RACE

- 클라이언트 객체의 재사용 문제
 - 고정 값이 더 이상 고정 값이 아니게 됨
 - 예) m_sock
- ID재사용 시 오동작 문제
 - 321번 플레이어에게 fire breath => 10초후 명중
 - 321번 플레이어 로그 아웃
 - 새로운 플레이어 A 로그인 321번 ID 부여
 - 321번 플레이어에게 fire breath 명중.
- 실제 게임 서버
 - ID 재사용 안함, new/delete로 Client객체 관리
 - 점검 시간에 ID 값 리셋

정리

- IOCP MMOG 서버의 뼈대 완성
- 완벽하지 않음
 - 멀티 쓰레드 버그 (
 - x, y 좌표
 - 최적화 필요
 - 너무 잦은 lock
 - 너무 큰 Critical Section
 - id 재사용
 - 오동작의 원인.
 - 등등등

정리

- 이후의 내용
 - 멀티쓰레드 한번 더
 - 성능 측정
 - 컨텐츠 구현
 - Quest, Item, Skill, 전투....
 - DB에 연동되는 것 말고 별다른 것 없으므로 PASS
 - 시야
 - 성능, 맵핵 방지
 - 충돌 처리
 - 해킹 방지
 - AI
 - 몇 십만 마리의 몬스터..., 스크립트 연동