

## 효율적인 온라인 게임 서버를 위한 객체풀링 기법에 관한 연구\*

김혜영<sup>○</sup>, 함대현\*, 김문성\*\*

홍익대학교 게임학부 게임 소프트웨어 전공<sup>○</sup>, 홍익대학교 게임학부\*

미시건대학교 컴퓨터공학부\*\*

hykim@hongik.ac.kr, gkaeogus0925@naver.com, mkim@msu.edu

### A Study of Object Pooling Scheme for Efficient Online Gaming Server

Hye-Young Kim<sup>○</sup>, Dae-Hyeon Ham\*, Moonseong Kim\*\*

Major in Game Software<sup>○</sup>, School of Games\*, Hongik University

Computer Science and Engineering, Michigan State University, USA\*\*

#### 요 약

대부분의 온라인 게임서버 엔진에서의 Accept()사용한 Looping방식의 동적인 메모리 할당 방식의 사용은 다수의 클라이언트를 동시에 수용하고 진행해야하는 온라인 게임 상의 로그인 서버엔진에 많은 부하와 함께 병목현상을 유발하게 된다.

따라서 본 논문에서는 온라인상에서의 효율적인 게임서버를 위해 정적인 메모리 할당기법을 사용하여 메모리 단편화를 최소화하고, 클라이언트의 접속 시 세션연결 및 클라이언트 객체의 초기화를 위해 발생하는 부하를 최소화하기 위한 AcceptEx()를 사용한 풀링기법과 정적메모리를 생성하여 동적 할당처럼 포인터로 할당하는 메모리 풀링 기법을 합친 객체 풀링기법을 제안하고, 제안 기법을 적용한 게임엔진을 설계하여 구현한 후, 성능평가를 통해 제안한 기법의 효율성을 보였다.

#### ABSTRACT

There is a request from the client, we almost apply dynamic memory allocating method using Accept(of looping method); thus, there could be process of connecting synchronously lots of client in most of On-line gaming server engine. However, this kind of method causes on-line gaming server which need to support and process the clients, longer loading and bottle necking.

Therefore we propose the object pooling scheme to minimize the memory fragmentation and the load of the initialization to the client using an AcceptEx() and static allocating method for an efficient gaming server of the On-line in this paper. We design and implement the gaming server applying to our proposed scheme. Also, we show efficiency of our proposed scheme by performance analysis in this paper.

**Keyword** : game server, object pooling, session

접수일자 : 2009년 07월 22일

일차수정 : 2009년 09월 21일

심사완료 : 2009년 10월 16일

\* “이 논문은 2007년 정부의(교육인적자원부)의 재원으로 한국학술진흥재단의 지원을 받아 수행된 연구임” (KRF-2007-531-D00011)  
또한 “이 논문은 2009학년도 홍익대학교 학술진흥비에 의하여 지원되었음”

## 1. 서론

온라인 게임은 네트워크상에서 하나이상인 다수의 클라이언트가 통신망을 통해 서버에 접속하여 실시간으로 진행되는 형태의 게임이라 할 수 있다 [1]. 따라서 온라인 게임 상에서 원활한 게임을 진행하기 위해서는 다수의 클라이언트를 수용하고, 속도와 안정성 및 신뢰성과 보안성을 보장할 수 있어야 한다[2].

대부분의 온라인 게임 서버 엔진에서는 클라이언트의 접속요청의 발생 시 동적인 기법을 적용하여 세션을 연결하고 초기화하는 등의 프로세스들이 진행된다. 하지만 이러한 기법은 다수의 클라이언트를 동시에 수용하고 진행해야하는 게임서버 엔진에 많은 부하와 함께 병목현상을 유발하게 된다[3,4].

따라서 본 논문에서는 효율적인 온라인 게임서버, 특히 로그인 서버에서 AcceptEx()를 사용한 풀링기법과 정적메모리를 생성하여 동적 할당처럼 포인터로 할당하는 메모리 풀링 기법을 합친 객체 풀링기법을 제안하고, 제안 기법의 효율성을 보였다.

본 논문의 구성은 다음과 같다. 다음 장에서는 효율적인 온라인 게임 서버를 위해 제안하는 배경 및 기법을 자세히 설명하고 3장에서 제안 한 기법을 적용한 온라인상의 게임 서버의 설계 및 구현에 대해 나열하고, 4장에서는 구현한 게임서버와 기존의 온라인 게임서버의 성능 분석을 통해 제안 기법의 효율성을 보였다. 마지막 부분은 향후 연구 방향 및 결론으로 구성하였다.

## 2. 제안기법

### 2.1 제안배경

대부분의 온라인 게임 서버에서는 각 클라이언트와의 통신을 담당하기 위한 소켓과, 비동기식 I/O에서 송수신시에 사용자 데이터의 데이터를 복사를 위한 버퍼가 필요하다. 그런데 접속 시마다 이러한 소켓 및 버퍼를 생성하고, 종료 시마다 삭제

하는 동적인 기법을 사용하게 되면 소켓을 생성할 때의 부하와 힙 메모리를 할당할 때 생기는 부하를 무시 할 수 없게 된다. 왜냐하면, 온라인 게임 서버에서 수 백, 수 천 개의 클라이언트가 생성되고 삭제되므로 게임서버에 접속하는 클라이언트 객체와의 연결을 위한 소켓의 생성 및 삭제가 빈번하게 일어나게 되며, 여러 번의 동적 할당으로 인한 메모리 단편화가 발생하기 때문이다. 이는 게임 서버에 많은 부하와 병목현상을 유발하게 되며, 온라인 게임서버의 성능을 저해하는 중요한 요인이 된다[5].

따라서 예상되는 필요만큼의 소켓이나 메모리를 미리 정적으로 할당하여 사용하고 이 후에 재활용하는 정적인 기법을 적용하여, 각 클라이언트와 통신하는 세션과 클라이언트가 필요로 하는 메모리 등의 리소스를 통합하여 하나의 객체로 설계하는 객체 풀링 기법을 사용함으로써 소켓과 버퍼를 재사용함으로써 대량의 동시 접속과 추가삭제가 빈번한 로그인 서버의 부하를 줄임으로써 온라인 게임 서버 성능의 효율성을 올릴 수 있다.

### 2.2 객체풀링기법

윈도우 운영체제상에서의 온라인 게임서버에서의 객체 풀링을 위해서 WinSock 확장 함수를 사용하는데 AcceptEx( )와 TransmitFile( ) 함수문을 사용하여 소켓을 재사용한다.

소켓 재사용함수를 적용한 객체 풀링하는 기법은 IOCP (Input Output Control Protocol)에서 사용하는 쓰레드 풀링 (thread pooling)처럼, 미리 만들어서 접속이 된 이후에 활성화되기 전까지 대기할 때 쓰이는 비사용 리스트와 (list)와 접속이 된 이후에 활성화되어 서버에서 필요에 따라 유니캐스트 (unicast), 멀티캐스트 (multicast), 브로드캐스트 (broadcast) 할 때 사용할 사용맵 (map)으로 구분하여 관리한다. 이러한 자료구조는 Session을 관리할 Session Manager클래스에 설계하였다. [그림 1]에서는 제안기법에서 사용하는 자료구조를 나타내었다.



[그림 1] 객체풀링기법의 자료구조

게임서버는 필요한 양만큼의 Session을 동적으로 생성한 후에 list에 삽입하고 listen 소켓을 IOCP에 연결한다. Accept( )함수의 비동기식 버전으로 접속을 요청한 후에 완료통보를 받고, 접속에 대한 작업을 수행하는 AcceptEx( ) 함수를 통해 비동기식 접속 요청을 한다. 이후에 워커쓰레드에서(worker thread) IOCP에 연결된 소켓을 통해 성공적인 접속 완료시, GetAcceptExSockaddrs( )를 통해 클라이언트의 주소를 받아오고 접속이 된

Session을 Session Manager의 비사용 리스트에서 검색 후에 사용 맵에 추가하고, 비사용 리스트에서 삭제한다. 또한 워크쓰레드를 통해 종료가 확인된 경우에는, AcceptEx( )함수에 쓰인 소켓을 재사용할 수 있도록 TransmitFile( ) 함수를 사용하여 소켓 재사용을 요청하고 Session Manager의 사용에서 검색하여 해당 클라이언트에 대한 리소스(resource) 지우고, 비사용 리스트에 다시 삽입 후 AcceptEx( ) 함수를 호출하여 다시 비동기식 접속을 요청한다. [그림 2]에서는 제안 기법을 도식화하였다.

제안 기법에서 적용한 객체풀링기법은 객체를 자체적으로 풀링하여 정적인 관리를 함으로써 클라이언트의 접속 시마다 발생하는 게임서버의 작업부하를 최소화하고, 메모리 리소스 관리를 Session 자체에 각 클라이언트와의 통신에 사용할 메모리를 원형 큐를 사용하여 적용함으로써 효율적인 풀링을 하였다.

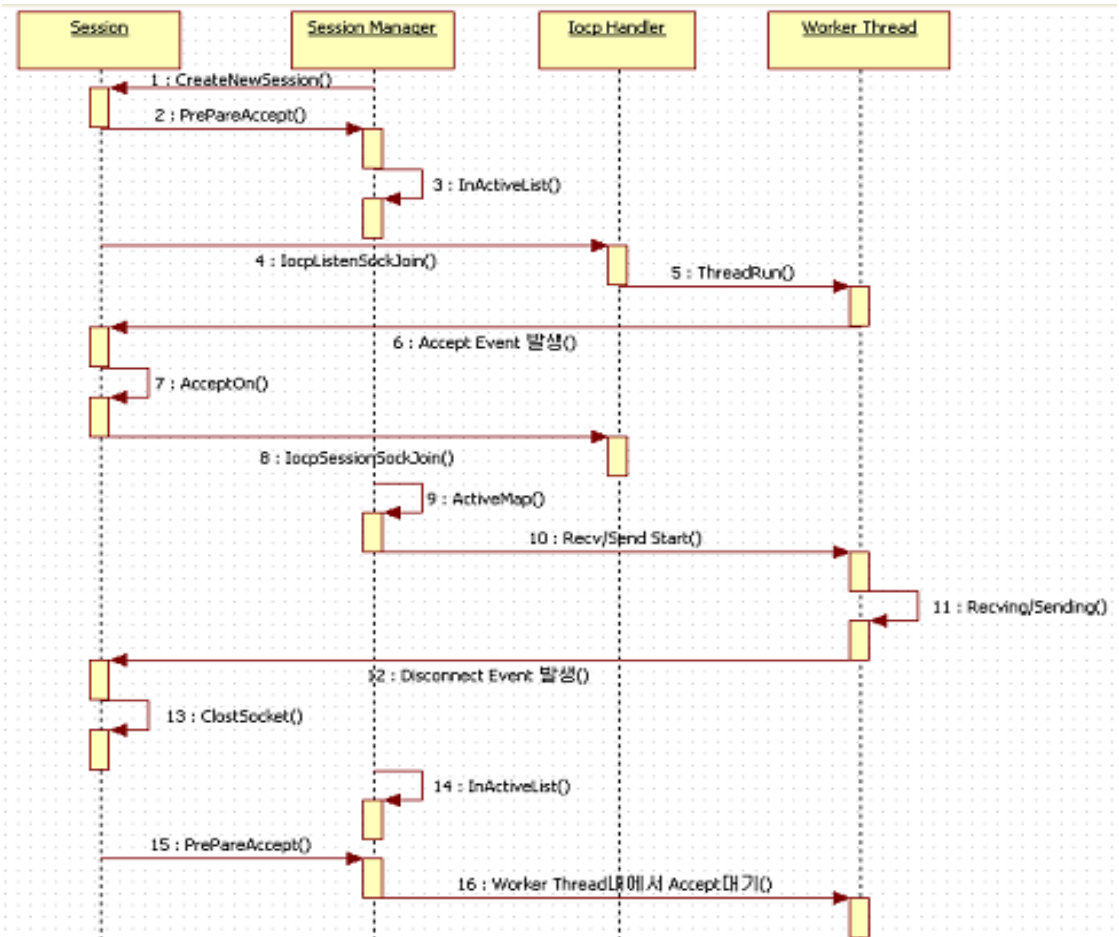
### 3. 제안 기법을 적용한 게임 서버 구현

#### 3.1 구현환경

본 논문에서 효율적인 온라인 게임서버를 위해 객체풀링기법을 적용하여 DirectX 9.0c SDK 2008 October를 이용하여 Windows XP 운영체제에서 게임서버를 구현하였다.

#### 3.2 게임 서버 구조

본 논문에서 제안한 기법을 적용할 게임서버의 구조도는 [그림 3]과 같다.



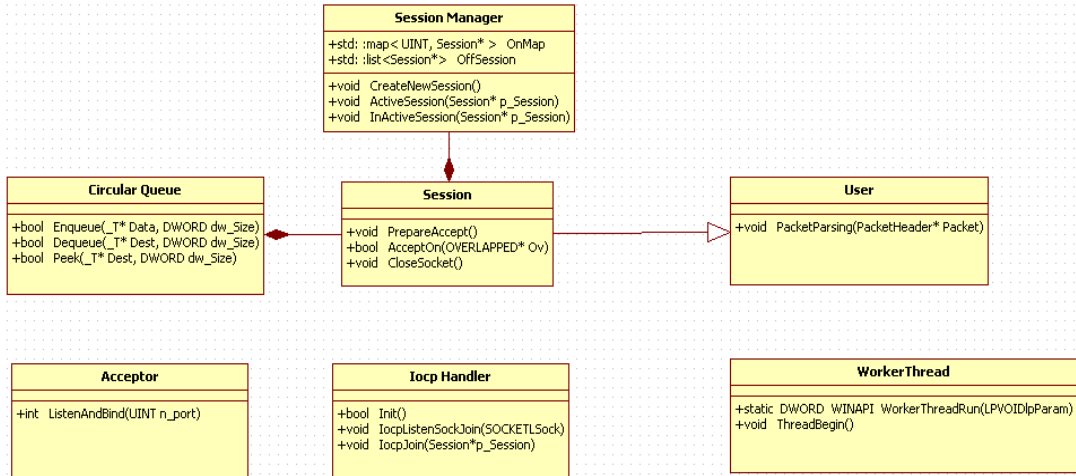
[그림 2] 제안기법

1. CreateNewSession()함수를 호출해서 동적으로 Session을 생성
  2. 생성한 Session에서 PrePareAccept()함수를 호출해서 AcceptEx()호출로 비동기 접속요청
  3. Session Manager에서 생성한 Session으로 InActiveList에 삽입
  4. ListenSocket을 IOCP Port에 연결시킴
  5. Thread를 생성하고 Worker Thread함수를 호출하여 GQCS()호출로 이벤트 대기
  7. Accept Event 발생시 Session의 주소를 받아오는 GetAcceptExSockaddrs()를 통해 주소를 받아옴
  8. AcceptEx()함수로 리턴된 Socket을 IOCP Port에 연결시킴
  9. AcceptEx()를 통해 접속이 된 Session을 InActiveList에서 삭제하고 ActiveMap에 삽입
  10. Recv/Send 작업 수행
  13. Worker Thread의 GQCS()에서 종료 Event 감지 시 종료가 감지된 Session을 shutdown()를 통해 송수신 작업을 종료시키고 TransmitFile()을 호출하여 소켓을 재활용시킴
  14. Session Manager의 ActiveMap에서 Session을 삭제하고 InActiveList에 다시 추가
  15. 종료된 Session을 다시 PrePareAccept()를 통해 비동기식 입출력을 요청
- 1~16번의 과정을 속 반복함으로써 객체를 재사용한다.

[그림 3]에서의 Session 클래스는 PrepareAccept( ) 함수를 사용하여 Session에 존재하는 Client 소켓과 Listen 소켓을 사용하여 AcceptEx( )를 호출하여 비동기식 접속 요청을 수행하고 요청을 수행하며, AcceptEx( ) 후에 주소를 받아오기 위한 GetAcceptExSockaddr( )를 호출하여 주소를 받아오는 작업을 수행하기 위해 AcceptOn(OVERLAPPED\* Ov)을 사용하였다. 또한 CloseSocket( )에서는 Session 클래스를 종료하고 재활용하기 위해 전송을 종료하는 shutdown( )를 사용한 후에

transmitFile( )로 소켓을 재활용 시켰다.

Session Manager클래스에서는 정해진 Session의 객체 수만큼의 동적인 생성을 통하여 OffSession(List)에 추가하고 Session클래스의 PrepareAccept( ) 함수를 불러 비동기식 접속 요청을 하도록 CreateNewSession( )을 구현하였다. 또한 Session 클래스에서 전달인자로 받은 Session 클래스를 OffSession(List)에서 삭제한 후에 OnMap(Map)에 추가하기 위해 ActiveSession(Session\* p\_Session) 설계를 했다.



[그림 3] 제안기법을 적용한 게임서버의 구성도

Iocp Handler 클래스에서는 Init( ) 부분에서 CreateIoCompletionPort( )를 사용하여 IOCP 포트를 생성하고 IocpListenSockJoin(SOCKET Sock)에서 소켓 값으로 IOCP포트에 전달된 소켓의 연결을 위해 CreateIoCompletionPort( )를 사용하였다 또한 접속 완료 후, 접속된 Session을 전달인자로 받아 IOCP포트에 연결하기위해 CreateIoCompletionPort( )를 사용하여 IocpJoin (Session\* p\_Session) 메서드를 구현했다.

Acceptor 클래스에서는 ListenAndBind (UINT n\_port) 메서드에서 전달된 포트번호로 Listen( ) 및 Bind( )를 수행하며 Listen 소켓을 관리한다[6].

온라인 게임서버를 위해 제안한 객체 풀링을 적용하여 구현한 온라인 게임서버 엔진의 중요한 부분에 대해 아래와 같이 간단한 Pseudo 코드로 설명하였다.

```

- Listen소켓을 IOCP포트에 연결
  CreateIoCompletionPort((HANDLE)m_ListenSock,m_HandleIocp,(ULONG_PTR)0,0);
- 객체 생성 후 AcceptEx( )로 비동기 접속 요청 후 비사용 리스트에 추가하기.
for(int i=0;i<객체를 풀링할 개수;i++)
{
    Session* p_Session = new Session()

```

```

Session->AcceptEx() //AcceptEx()함
        수로 비동기식 접속 요청
    InActiveList.push_back(p_Session)
        //비사용 리스트에 Session추가
}
- IOCP 완료 큐를 감지하는 Worker Thread에
  서 접속 확인 시 비사용 리스트에서 삭제 및
  사용 맵에 추가

* 비사용 리스트에서 삭제
for( std::list<Session*>::iterator iter =
InActiveList.begin(); iter != InActiveList.end();
{
    if(*iter == p_Session)
    {
        ActiveMap.insert(소켓값, Session 포인터);
        iter= InActiveList.erase(iter);
    }
    else
    {
        iter++;
    }
}

* 사용 맵에 추가
std::map<UINT,Session*>::iterator iter =
OnMap.find(p_Session->GetSocketValue());
if( iter != OnMap.end() )
{
    InActiveList.push_back( p_Session );
    iter = ActiveMap.erase( iter );
}
- IOCP 완료 큐를 감지하는 Worker Thread에서
  접속 종료 시, 클라이언트의 정보를(resource)
  사용 맵에서 삭제 후 TransmitFile( )를 사용
  하여 소켓 재사용 요청 후 비사용 리스트에
  다시 추가하고, AcceptEx( )를 다시 호출한다.
shutdown( m_SessionSock, SD_BOTH );

```

## 5. 성능평가

본 논문의 3장과 4장에서 언급한 제안 기법을 적용한 온라인 게임 서버 엔진의 성능 비교를 위해 세션 수의 증가 시의 수행속도를 기존의 게임 엔진과 비교하였다. 이를 위한 게임서버의 테스트 환경은 [그림 4]와 같다.

```

시스템:
Microsoft Windows XP
Professional
Version 2002
Service Pack 2

사용자 정보:
팜
76489-640-0765433-23340

컴퓨터:
Intel(R) Core(TM)2 Duo CPU
T9300 @ 2.50GHz
772MHz, 3.00GB RAM

```

[그림 4] 게임 서버 테스트환경

본 논문에서의 제안 기법을 적용한 게임서버의 테스트를 위해 간단한 방식의 이동 및 박스와 충돌, 그리고 채팅이 가능한 클라이언트를 제작하였으며, DirectX9.0C를 사용하여 클라이언트 측에서는 EventSelect( )함수를 사용하여 소켓 이벤트를 감지하여 패킷을 수신하여 분석하였으며, DirectX에서 필요로 하는 좌표 float값 x, y, z와 시야각 float값 ViewDis 및 ID, 그리고 채팅에 필요한 char형을 서로 주고받도록 제작하였다. [그림 5]에서는 클라이언트의 실행화면을 보여준다. 기존 Accept방식에서는 동시에 접속할 수 있는 최대 세션 수는 Listen함수가 허용하는 5개이고, 이를 늘리기 위해서는 많은 양의 쓰레드가 필요하다[7]. 하지만 IOCP (Input Output Control Protocol)에서의 AcceptEx()의 경우에는 적은 양의 쓰레드를 가지고 미리 생성한 객체 수만큼은 동시에 받을



클라이언트의 접속 시 세션연결 및 클라이언트 객체의 초기화를 위해 발생하는 부하를 최소화하였다.

또한 제안한 기법을 적용한 게임서버를 위한 게임엔진을 설계하고 구현하였으며, 성능평가를 통해 제안한 기법의 효율성을 보였다. 그러나 본 논문에서 제안한 기법에 대한 효율성을 높일 수 있는 온라인 게임 상의 클라이언트 측의 정적 할당에 대한 최적화된 값인 스레시 홀드 값(threshold value)을 추정해내지 못하고 256개의 소켓과 4kbyte의 메모리를 정적 할당 시에 적용하였다. 따라서 향후 연구에서는 온라인상의 게임서버에 접속하는 세션에 대한 정확한 예측을 위한 분석모델을 설계하고 수학적 접근에 의한 모델링을 통해 클라이언트 측의 정적 할당에 대한 최적화된 값을(threshold value) 제안함으로써 더욱 효율적이고 신뢰성 있는 온라인 게임 서버 엔진에 대해 연구하고자 한다.

## 참고문헌

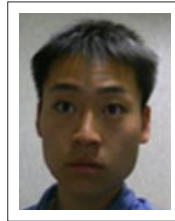
- [1] 김선우, 윈도우 네트워크 프로그래밍, 한빛미디어, 2007
- [2] 이상원 김혜영, 게임엔진에서의 효율적인 메시지 관리 기법, 한국 게임 학회 논문지, 제 8권 제2호, 2008년 5월
- [3] Michael Duck and Richard Read, Data Communications and Computer Networks, Prentice Hall
- [4] James F. Kurose and Keith W. Ross, Computer Networking, Addison Wesley
- [5] 한동훈, 온라인 게임 서버 프로그래밍 벤치마크, pp.337~340, 정보문화사, 2008
- [6] 한동훈, 온라인 게임 서버 프로그래밍, 정보문화사, 2007
- [7] 엄남경, 문형진, 이상호, 게임서버간 부하의 균일성에 기반한 부하 분산기 설계 및 구현, 한국통신학회논문지, Vol.32, No.3, 2007
- [8] J.Chen, B.Knutsson, B.Wu, H.Lu, M.Delap and C.Amza, Locality Aware Dynamic Load Management from Massively <multiplayer Games, PPOPP, 2007



김혜영(Hye-Young Kim)

2005년 2월 고려대학교 컴퓨터학과 이학박사  
2005년 3월~2006년 8월 Wright State Uni. Post-Doc.  
2007년 3월~현재 홍익대학교 게임학부  
게임소프트웨어전공 조교수

관심분야 : 모바일 게임, 온라인 게임서버, 게임엔진



함대현(Dae-Hyeon Ham)

2003~ 홍익대학교 재학

관심분야 : 게임 엔진 구조, 온라인 게임서버  
프로그래밍, 네트워크 프로그래밍



김문성(Moonseong Kim)

2007년 2월 성균관대학교 전기전자 및 컴퓨터공학과  
졸업, 공학박사  
2007년 12월~현재 Visiting Research Associate,  
Michigan State University, USA

관심분야 : 모바일/온라인 게임, 모바일컴퓨팅, 라우팅  
프로토콜