Overlapped IO와 IOCP 이야기

이 기탁 (Microsoft 2002 Asia MVP)
E-Mail: snaiper80@korea.com ,
nsnaiper@hotmail.com(MSN ID)
Devpia ID: snaiper

◆ 시작하기 전에

이것은 제가 정리하기 위한 글이기도 한 강좌입니다. 뭐 강좌라 하기도 그렇군요 ^^: 강좌라 하기에는 조금 모자랄 걸로 보입니다. 일단 IOCP 에 대한 전반적인 정리를 해 볼 생각입니다. 그리고 코드 보다는 개념 쪽에 초점을 맞출 겁니다. 그래서 예제는 기대하지 마시길 ^^ 뭐 현재 계획으로는 그렇습니다만 글쎄요. 앞으로 달라질 수도 있을 수도 있겠네요.^^: 일단 얼마나 길어질지 아니면 짧아질지는 잘 모르겠습니다. 처음이다 보니 좀 부족하니까 대충 계획 잡으니 어떻게 될지 알 수 없네요. 뭐 해보는 데로 해보죠.(무대포가 되겠군요 ^^) 아 그리고 참고자료는 제가 개인적으로 가지고 있는 자료와 제프리 책들입니다. (이 책들은 윈도우 프로그래머에게는 거의 필수죠?)

아 그리고 마지막 말씀 드리고 싶은 것은 저도 아직 초보이고 하니까 분명히 틀린 것이 있을 겁니다. 모자란 점도 있을 거고요. 그런 점은 지적해 주시면 감사하겠습니다.

◆ Overlapped IO와 IO Completion Port의 관계

IOCP 얘기를 하기 전에 Overlapped IO 에 대한 지식은 필수적입니다. IOCP는 이 Overlapped IO를 바탕으로 해서 통보를 받아 처리하는 방식 중의 하나입니다. 아마 관심 있으신 분들을 보셨을 겁니다. IOCP 예제를 보시면 WSARecv나 WSASend에 반드시 Overlapped 구조체를 집어넣죠? 그리고는 GetQueuedCompletionStatus() 에 보면 4번째 인자로 Overlapped 구조체가 넘어오죠? 자 이걸 보면 이걸 이해하지 않고 넘어가는 건 왠지 안 될 것이라는 건 감이 오실 겁니다. 안 오신다고 하신다면 흠 어쩔 수 없죠 ^^: 그리고 한번도 이 함수를 안 보신 분이라면 MSDN 한번 찾아보세요. 그런데 혹시나 MSDN에서 어떻게 찾냐고 하는 분은 없으시겠죠? 지금 집고 넘어 가는데 이걸 읽으시는 분들은 적어도 어느 정도는 아시는 초초보 단계는 때셨다고 가정합니다. 하기야 이거에 관심 가지실 분들은 이미 초초보의 단계는 충분히 넘으셨을 겁니다.

◆ Overlapped IO

위에서 왠지 중요한 것이라고 말한 Overlapped IO 가 뭐냐? 도데체 뭐하는 놈인데

그러냐? 하실 겁니다. 자 그래서 이걸 설명하죠.

◆ Overlapped IO란?

넌블러킹이고, 비동기적으로 IO를 처리하기 위한 IO 방법입니다. 자 이러면 별로 감이 안 오실 것 같군요. 비동기적이라는 것이 왜 중요하냐? 이것부터 시작해야겠네요.

컴퓨터 써보신 분들(이거 읽으시는 분들 중에 여기 포함 안 되시는 분들은 없겠죠?)은 아시겠지만 플로피 디스크나 하드 디스크는 CPU보다 느립니다. 그래서 게임할 때 느린 로딩 속도 탓하면서 아 내 컴퓨터는 왜 이렇게 느리지 하는 것도 IO가 느려서 그렇죠. 그래서 하드디스크는 좋은 걸 사야 한다는 말씀입니다. ^^ 아 본론은 이 얘기가 아니죠. 이 IO라는 것은 서버와 같이 IO를 대량으로 처리하는 곳에서는 아주 크리티컬한 요소가 됩니다. 그래서 만약에 이 IO를 처리한다고 쓰레드가 하염없이 기다리고 있으면 이 서버에 접속하는 사용자들 왜 이렇게 느리냐고 난리 날 겁니다. 아 왜 기다리느냐 안 기다리면 되지 않느냐 하겠죠? 예 그렇습니다. 안 기다리면 되죠. 하지만 기본적으로 IO는 동기입니다. 즉 ReadFile 이렇게 하면 지정해준 파일 읽기 작업이 끝날 때 까지는 함수가 리턴되지 않습니다. 이런걸 보고 블러킹 되었다고 하죠. 이럼 쓰레드는 니나노 하면서 놉니다. 조금 어렵게 말하면 suspend 상태에 있다고 할 수 있겠네요. 그럼 이 시간 동안은 어떤 작업 처리도 못하고 있는 거죠. 그래서 앞에서 말한 것처럼 안 기다리는 작업이 필요해진 겁니다. 그래서 나온 게 비동기적으로 IO처리하는 것입니다. 뭐 이럼 CPU도 안 놀리고 하니 자원 활용 측면에서도 좋겠죠? 그럼 비동기로 IO 어떻게 처리하는데? 라고 물으실 수 있겠네요. 이미 알고 계실 걸로 믿지만 그게 Overlapped IO 라는 겁니다.

◆ Overlapped IO의 작동

자 그럼 이 Overlapped IO가 어떻게 작용하는 것인지가 궁금해지실 겁니다. 그걸 한번 생각해보죠.

일단 IO를 하나 해야 Overlapped IO 가 일어나든 말든 하겠죠? 그럼 네트웍으로 패킷 하나 보낸다고 합시다. 그럼 WSASend(어쩌구 저쩌구…., lpOverlapped,..)
뭐 이런 식의 코딩이 되겠죠? 그럼 Overlapped 구조체가 지정되어 있으니까 이것은 바로 리턴됩니다. 아마 리턴값은 SOCKET_ERROR 이고 에러 이유를 WSAGetLastError() 로 살펴보면 ERROR_IO_PENDING 뭐 이런 거 일겁니다. IO요청이 성공했다는 거죠. 그리고 지금 작업 중이니 신경 끄세요. 라는 말입니다. 뭐 이런 경우가 아닐 수도 있긴 합니다만…… 제 경험상 대부분 이거더군요.

그건 그렇고 약간 빗나가는 얘기지만 소켓으로 IO 하든 파일 IO 든 간에 반드시 에러 처리는 해주셔야 합니다. 가끔 안 해주시고 이게 왜 안되죠? 하고 질문하는 분들

이 있더라고요. 그럼 답변하는 사람은 그 사람 컴퓨터를 만져본 게 아니니까 당연히 모르겠죠? 그래서 꼭 해주세요. 안 해주시면 안됩니다. 꼭 리턴값 확인해보시고 에러 나면 GetLastError, WSAGetLastError 함수로 뭔 에러인지 알아보세요. 아니면 나중에 크게 고생하실 겁니다.

자 빗나가는 얘기는 넘어가고요. 본격적으로 다시 Overlapped로 넘어와야죠. 앞에서처럼 IO를 해주면 어떻게 되느냐? 그럼 이 비동기적으로 IO를 요청했다는 정보가 Device에게 날아갑니다. 뭐 더 정확하게 말하면 Device Driver 입니다. 그럼 이 IO처리 책임은 이 Driver 에게 넘어가는 거죠. 더 이상은 우리 쓰레드 군이 신경 쓸 것이 아니라는 겁니다. IO 처리는 그 놈이 열심히 해줄 것이고 쓰레드는 열심히 딴 일을 처리하는 겁니다. 뭐 다른 IO도 또 Device Driver 에게 요청하거나 열심히 for 문 돌아주거나 하는 이런 처리 하겠죠? IO 끝날 때까지 하염없이 기다리는 것 보단 CPU도 놀리지 않고 훨씬 유용한 작업이 될 겁니다.

자 그럼 IO가 끝날 때를 살펴보죠. Device Driver 는 열심히 IO 하는 걸 감시하면서 처리하고 있다가 끝나면 우리한테 알려줍니다. 안 알려주면 우리는 끝난 건지, 아님 자는 건지, 에러 나서 실패한 건지 알 수가 없겠죠? 자 여기서 우리한테 알려주는 방법이 중요한데 여기서 IOCP 가 짠!!! 하고 등장하는 겁니다. 일단 IOCP보다는 딴 것을 이야기 해보죠. 원래 Overlapped IO를 이용하여 우리에게 통보해주는 방법으로는 여러 가지가 있습니다. OVERLAPPED 구조체에 있는 이벤트 객체 멤버를 이용해서 이거 가지고 신호하는 방법, 아니면 지정해주는 Callback 함수 가지고 이걸 호출해서 알려주는 방법, 아니면 사용자가 직접 묻는 방법 이렇게 있죠.
이건 나중에 얘기하겠습니다. (이러고 까먹으면 큰일이죠? ^^:::)

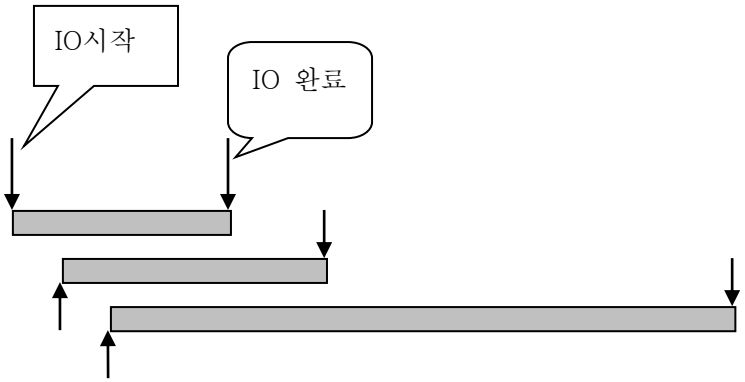
일단 중요한 건 IOCP 얘기니까 이걸 얘기해보죠. 위에서 얘기한 방법들은 다 IO를 요청한 쓰레드에서 처리하는 방법입니다. 이것도 나쁘지야 않겠지만 이 처리마저 전담하는 다른 놈에게 맡기고 나는 유용한 처리를 계속 하고 있으면 더욱 좋겠죠. 그래서 나오는 것이 IOCP입니다. 이건 IO 완료 처리 전담하는 놈이 몇 명 있어서 IO를 요청한 놈은 신경 쓰지 않고 딴 놈이 뒷 처리를 해주는 겁니다. 이러니 곰은 재주 넘고 돈은 사람이 챙긴다는 말이 생각나는군요 ^^ 그래서 이렇게 하면 가장 성능이 좋더라 해서 나온 것이 MS의 연구 결과의 산물인 이 IOCP 이죠. 실제로 이런 것을 Concurrent Pattern 에서는 Proactor Pattern 이라고 하더군요. 관심이 있는 분은 ACE라는 것을 찾아보세요.

보자 이럼 IO 요청하고 IO 완료 결과 받는 것 까진 끝난 거네요. 이럼 설명이 제대로 된 건지 모르겠습니다. 잘 이해가 되셨을런지…모르겠군요.

◆ Overlapped ?

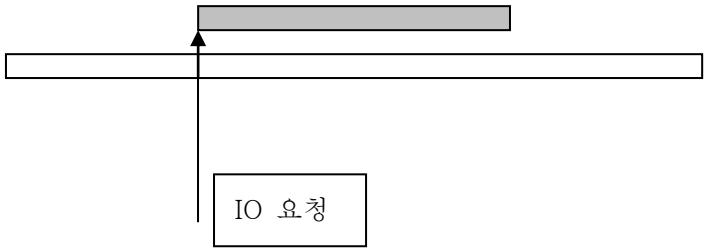
이제 그럼 Overlapped IO 에서 이 Overlapped 라는 것이 뭐냐고 궁금해 하실 분이 생기실 걸로 보입니다. 저도 정확한 정의가 없어서 여기저기 찾아보기 했는데 그리 만족스럽게는 알지 못했습니다. 뭐 여하튼 제가 이해한 걸 말씀 드리죠.

일단 첫 번째 입니다. 이 Overlapped 라는 것이 말 그래도 IO가 여러 개 중첩 되어서 처리된다는 얘기입니다. 그림으로 보면 이렇게 되겠네요.



이렇게 IO가 여러 개 중첩되어서 진행된다는 의미 입니다.

그리고 두 번째로는 자기가 하는 작업과 IO가 동시에 진행된 다는 의미죠. 이건 그림으로 그리면



Jeffrey Richter의 Programming Server-Side Programming 에서는 이렇게 설명되어 있습니다. 하지만 제가 가지고 있는 또 다른 자료에는 저렇게 설명되어 있네요. 어느 것이 정확한지는 저도 확실치 않습니다. 다음에 정확히 알면 말씀 드리겠습니다. 이 강좌를 읽어보시는 분 중에 정확히 아시는 분 있으면 알려주시길 바랍니다.

◆ 정리해볼까요?

자 일단 정리해보죠. 정리해보고 넘어가는 것이 좋겠죠?

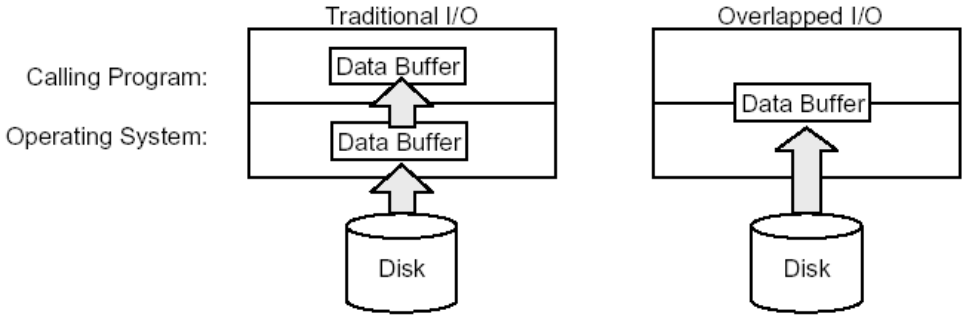
1. Overlapped IO는 넌블러킹, 비동기 IO를 위한 방법이다.
2. IO를 요청하고 기다릴 필요가 없이 리턴하고 딴 일을 처리한다.
3. 나중에 이 작업 완료 여부를 체크할 수 있는데 이 방법 중에 하나가 IOCP 이다.
4. 멀티로 IO 요청이 가능하다. (이건 정확히는 언급안한 것 같은데 뭐 정리하자면 그렇게 되겠죠?)
5. 자 가장 중요한 이걸 쓰는 목적! 성능을 위해서다. 가 되겠죠.

아 그리고 다음으로 넘어가지 전에 참고하실 점! 이 Overlapped IO는 Network IO인 소켓 recv, send 말고도 File IO에도 적용됩니다. 단 윈도우 9X 계열에서는 약간 제한이 있습니다. 뭐 소켓 프로그래밍하는 데는 지장 없으니 신경 안 쓰셔도 될는지도 모르겠네요. 그리고 나중에도 언급하겠지만 IOCP 는 네트웍만을 나온 전용이 아님을 알아주셨으면 합니다. 그리고 IO외에도 쓸 수 있는 가능성도 있습니다. Interthread Communication 에도 쓸 수 있습니다. 저도 이걸로 한번 써봤는데 귀찮게 Queue를 따로 만들지 않고 이걸 쓰니까 좋더군요. 큐에 또 여러 쓰레드가 붙을 수 있으니까 그것도 괜찮고요. 또 뭐 쓰레드 세이프한지 안 한지 신경 쓰지 않으니 좋은 듯 합니다.

◆ Overlapped IO의 좋은 점

자자 다음으로 넘어가죠. 위에서 Overlapped IO 에 대해 어느 정도 알아봤습니다. 이제 좀 더 뭐가 좋은지 짚어보죠.

이것의 좋은 점으로 또 한가지가 위에서 말한 IO Blocking 이 일어나지 않는다 와 버퍼링 오버헤드가 줄어든다는 점입니다. 이 Overlapped IO 는 IO를 처리할 때 사용자가 지정해준 버퍼를 바로 사용한다는 점입니다. 무슨 말이냐 하면 추가적으로 OS에서 제공한 버퍼를 사용하지 않고 직접 사용자 주소 공간에 있는 버퍼에다 직접 read/write 한다는 얘기입니다. 그래도 잘 이해가 안 되시면 그림이 최고겠죠?



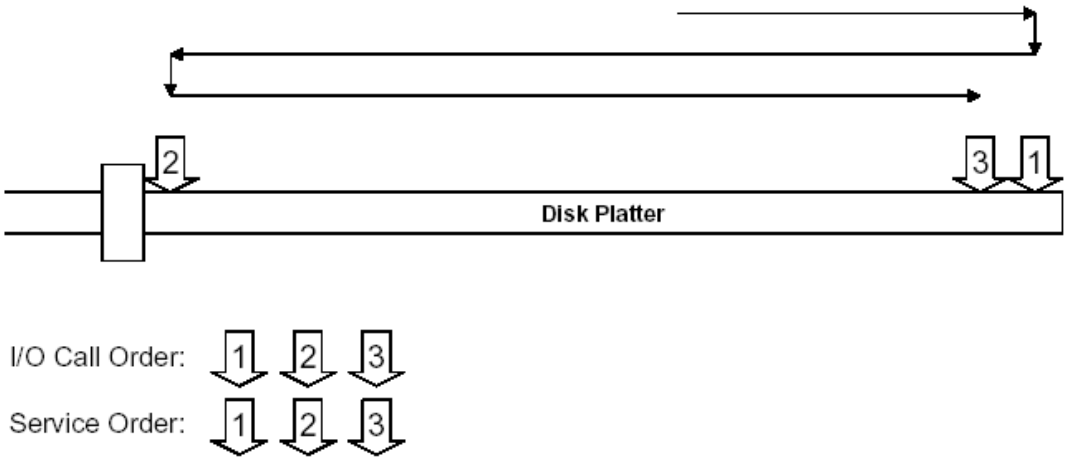
제가 가지고 있는 자료에서 가지고 온 그림입니다. (이렇게 무단으로 써도 되려나? ^^)

보시면 Disk IO 작업을 하는데 기존 IO는 OS에서 버퍼링하고 그 다음에 사용자가 지정해준 버퍼로 옮기는 것을 아실 수 있을 겁니다. 하지만 이 Overlapped IO는 보면 직접 카피되죠? 이럼 카피되는 작업이 하나 줄어드는 거죠? 이럼 더 성능이 좋아지겠죠?

하지만 이렇게 되면 예를 소켓 IO 작업이 되는 경우 Data Buffer 저것이 page lock 됩니다. 이것을 왜 언급하냐면 많은 IO 작업시에 문제가 될 지도 모른다는 거죠. Page lock 되면 디스크로 swap 될 수 없는데 이렇게 page lock 된 것이 일정량 이상은 불가능 하다는 말입니다. 그래서 나중에 에러가 나올 수도 있습니다. 뭐 버퍼가 모자라다. 아님 메모리가 모자라다 뭐 이런 식으로 말입니다. 조금만 더 자세히 얘기하자면 각각 Device Driver 들은 일정 사이즈의 IO요청 리스트를 가지고 있습니다. (이게 non-paged pool에 있습니다. 이건 말 그대로 절대 page될 수 없는 그러니까 하드로 swap 되지 못한다는 얘기입니다.) 그런데 이 사이즈가 고정되어 있기 때문에 이 사이즈 이상의 요청은 받아들이지 못한다는 얘기죠. 그 정확한 사이즈는 저도 정확히 모르겠네요. 언뜻 들은 바로는 이 non-paged pool의 크기가 physical memory 사이즈(컴퓨터에 꼽혀 있는 그거 말입니다 ^^)의 4분의 1이상은 넘지 못한다고 합니다. 이걸 가지고 계산해보면 될 듯한데 저도 확실히 모르겠습니다. ^^::

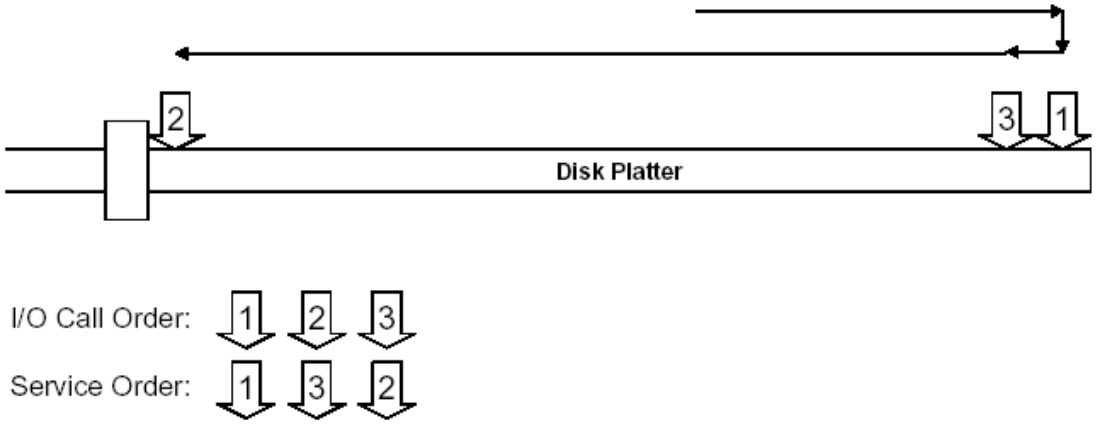
확실한 건 이런 문제가 나온다면 에러가 ERROR_INVALID_USER_BUFFER 나 ERROR_NOT_ENOUGH_MEMORY가 나온다는 거죠. 이건 ReadFile등의 파일 IO에 대한 경우이고 소켓 IO 같은 경우는 WSAENOBUFS 이게 나올 거라고 들었습니다. 그런데 이거 부딪칠만한 일이 거의 없으니 알 수가 없군요. 진짜 수많은 IO가 나와야만 볼까 말까 하는 것이거든요. 이건 나중에 알아보고 말씀 드리죠. 역시 아시는 분 있음 저한테 말씀해주세요 ^^ (이런 무책임한 --;)

자 그리고 다른 걸 보죠. Overlapped IO의 특징 중에 하나가 FIFO로 IO를 처리하지 않는다는 겁니다. 그럼 그림을 보죠.(이제 그냥 그림 가지고 올랍니다. 그림 그리기가 귀찮네요 ^^: 그림이 너무 잘 그려져 있으니 굳이 제가 안 되는 그림 실력으로 그릴 이유가 없네요 ^^:)



자 기존의 방식은 이렇게 디스크를 왔다 갔다 하면서 FIFO 방식으로 처리합니다. 그러니까 IO 요청 들어온 데로 왔다리 갔다리 한다는 거죠.

하지만 Overlapped IO는 조금은 비효율적인 이런 스케쥴링 방법을 쓰지 않습니다.
그게 뭐냐면 SCAN 방식입니다. 이건 운영체제 책에도 나옵니다. 책 가지고 계신 분은 한번 살펴보세요. 다음 그림을 보세요



여기 보면 IO 요청이 1,2,3 이렇게 되어도 작업 처리는 1,3,2 이런 식으로 되는 걸 볼 수 있습니다. 그러니까 한 방향으로 쭉 가면서 그 근처에 있는 걸 처리하고 간다고 볼 수 있죠. FIFO방식에서는 이 쪽 왔다 갔다 하면서 처리했습니다. 하지만 이러면 IO 속도가 떨어지죠. 왜냐고요? 디스크 헤드 움직이는 건 기계 적인 일인데 이걸 많이 하면 속도 떨어지는 건 당연하겠죠? 그러니까 이걸 최소화해야죠.

그리고 또 있는 것이 RAID시에 병렬적으로 IO를 처리할 수 있다고 합니다 이걸 Device Level Parallelism 이라고 하는군요. 이건 제가 RAID 쪽을 잘 모르니 별로 말

씀드릴 거리가 없군요. ^^:::::: 다만 IO를 여러 disk array 에 한꺼번에 요청하고 결과를 한꺼번에 받는다는 것 정도로 만족해주세요. 위에서 안 지식으로 생각해보면 기존 IO는 disk array 중 하나의 disk 에다 요청하고 결과 받고 또 다른 disk에다 처리하고 하는 거니 좀 비효율적이다 그런 거겠죠?

자 일단 여기까지 해서 하나는 끝내죠. 많이 하면 저도 힘들고 또 보는 사람도 머리 아플 테고요. 다음에는 Winsock 에서 Overlapped IO하는 걸 살펴보고 IOCP 설명 쪽으로 넘어가도록 할 겁니다. 바빠서 또 언제 쓸는지 모르겠습니다. 일단 최대한 시간 나는대로 쓰겠습니다.

그럼 다음에 뵙죠.