예측로딩을 통한 온라인게임 부하 감소 기법

김용오
고려대학교 컴퓨터과학기술대학교 미디어공학과
e-mail : koyanghi@korea.ac.kr

A Technique for Network Load Decrease by Predicated Loading Method

Yong-O Kim
Dept. of Media Enginerring, Korea University

요 약

온라인 게임의 한 분야인 MMORPG 는 하나의 가상 공간에서 많은 캐릭터들이 자신의 역할을 수행하면서 게임을 즐기는 장르다. 많은 캐릭터들이 동시에 이동을 하거나 캐릭터가 대량으로 밀집된 지역으로 새로운 캐릭터가 이동해 올 경우에 생기는 부하가 클라이언트의 게임 프레임에 영향을 준다.

본 논문에서는 다른 지역으로 이동하면서 생기는 많은 데이터들을 미리 예측 로딩하여 지역과 지역간의 이동 시 소요되는 데이터 로딩에 따른 부하를 감소시켜 안정적인 온라인게임 플레이 환경을 유지시키는 방안을 제시한다

1. 서론

최근 온라인 게임의 주를 이루고 있는 장르는 대용량 Role-Playing 게임인 MMORPG(massively multiplayer online role-playing game)분야이다[1]. 수 백 명 이상의 캐릭터들이 가상환경에 접속하여 주어진 역할을 수행하는 RPG 게임은 대량의 Packet 을 처리 할 수 있는 서버를 필요로 함과 동시에 많은 양의 데이터를 한꺼번에 화면에 표현 할 수 있는 높은 사양의 클라이언트 PC 사양이 요구된다. 3D 텍스쳐와 오브젝트를 화면에 출력하기 위해서는 해당 파일을 동시에 혹은 순차적으로 가져와서 메모리에 생성시킨다. 메모리 적재에 필요로 하는 시간이 화면 출력을 위한 게임 프레임 시간보다 늘어나게 되면 일시적인 화면 멈춤 현상이 발생하게 된다. 그와 동시에 게임 프레임과 순차적인 순서로 불려지는 네트워크 Packet 도 처리되지 못하고 다음 게임 프레임에 한꺼번에 처리된다[2]. 게임 프레임의 일시적인 하락과 Packet 의 동시 처리는 온라인 게임의 질에 직접적인 연관 되어 있어 유저의 게임 충성도에 많은 영향을 끼친다[3].

본 논문에서는 동시에 많은 데이터를 메모리에 적재시켜야 할 시기를 미리 예측하여 로딩하는 기법을 도입하여 위에서의 문제점들을 해결하는 구조를 제시한다. 본 논문은 다음과 같이 구성된다. 2 장에서 일반적으로 사용되는 온라인 게임의 구조에 대해 알아 보고, 3 장에서는 예측 로딩 기법을 사용하는 서버 구조에 대해 알아 본다. 4 장에서는 기존 구성과 성능평가의 결과에 대하여 설명하고 5 장에서 결론을 맺는다.

2. 온라인게임 서버 구조

2. 1 Peer to Peer 구조

일정한 서버가 존재 하지 않으며 각각의 클라이언트들이 연결된 peer 들과 통신을 하는 구조다. 일반적으로 서버를 통한 전체적인 처리를 필요로 하지 않는 실시간 전략게임이나 액션게임에서 많이 사용되는 구조로 MMORPG 에는 적합하지 않다.

2.2 Client to Server 구조

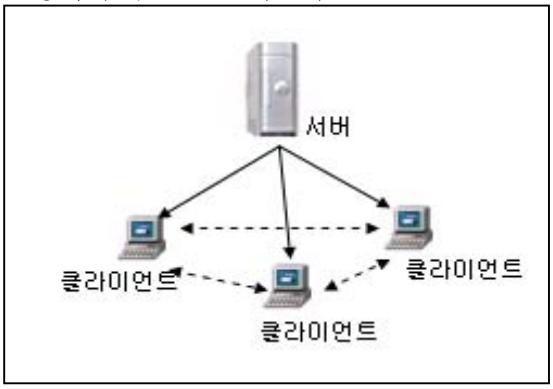
서버를 중심으로 각 클라이언트들이 서버와의 통신을 통해서 다른 클라이언트와 통신하는 방식으로 많은 온라인게임은 이 구조를 따른다.

2.2.1 중앙 서버 방식

하나의 서버에 모든 클라이언트가 접속된 구조로 한번에 많은 클라이언트의 작업을 처리하는데 발생하는 부하를 하나의 중앙 서버가 감당해야 한다.

2.2.2 하이브리드 방식

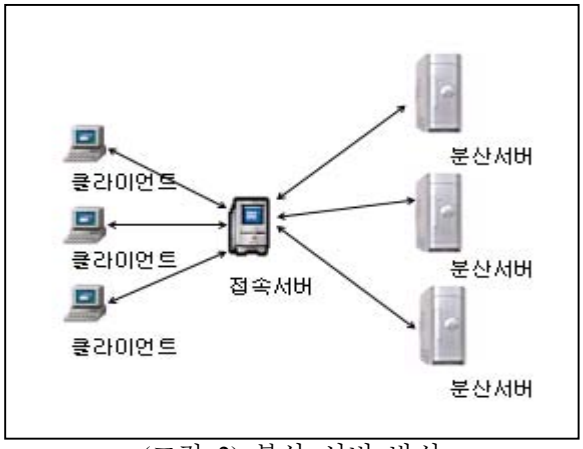
하나로 집중된 서버의 부하를 클라이언트들간의 통신으로 줄이는 방식으로 Peer to Peer 방식과 중앙 서버 방식의 혼합된 방식이다[4]. 서버가 담당해야 하는 전체적인 작업량은 줄어드나 클라이언트들 간의 통신에서 발생 할 수 있는 해킹 등의 문제를 추가로 해결하여야 하며, 처리해야 하는 작업을 Client To Server 와 Peer to Peer 로 나눠서 작업해야 한다. 그림 1 은 하이브리드 방식의 구조를 보여준다.



(그림 1) 하이브리드 방식

2.3 분산 서버 구조

일반적인 MMORPG 에서 많이 사용되는 구조로 중앙으로 집중된 서버를 분산 처리하는 구조다[5]. 그림 2 에서 중앙의 접속서버가 각 분산 서버의 부하를 판단하여 각 클라이언트들을 연결시켜 준다.



(그림 2) 분산 서버 방식

2.3.1 부하 분산 방식

웹 서버나 캐시 서버를 사용하여 처리의 부하를 균등하게 나눠 서버에서의 응답지연이나 처리 불능 등의 경우를 처리 량이 적은 다른 서버가 처리하는 방식이다.

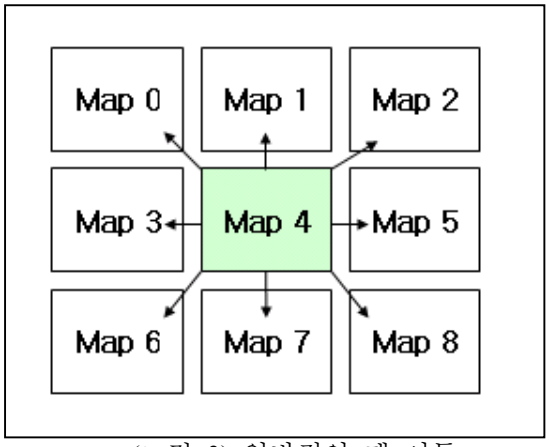
2.3.2 맵 서버 방식

게임 내부의 지역을 일정한 맵 단위로 분할한 뒤, 각각 맵 서버가 해당 지역에 접속한 클라이언트들의 처리를 담당하는 구조다. 본 논문에서는 맵 서버 방식을 사용한다.

3. 예측로딩을 통한 서버 부하 감소

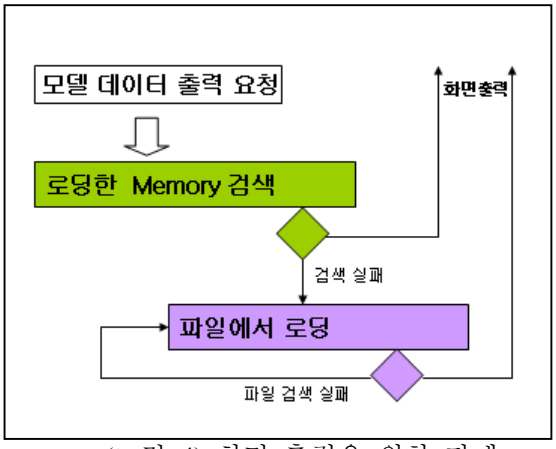
3. 1. 맵 서버 방식에서 맵 이동 처리

일반적으로 한 맵에서 다른 맵으로의 이동은 맵의 경계지역에서 이뤄진다. 맵의 크기와 모양은 게임의 특징에 따라 달라 질 수 있으나 본 논문에서는 크기와 모양의 일관성을 이루기 위해서 직사각형 형태로 맵을 형태를 정의한다. 일반적인 맵 서버 구조에서 각각의 맵에서 다른 맵으로 이동하기 위한 경우의 수는 8 방향 중 하나가 된다. 그림 3 에서 Map4 에 위치한 캐릭터는 Map4 를 제외한 8 개의 맵으로 이동이 가능하다. 클라이언트가 화면에 3D 텍스쳐나 오브젝트를 출력하기 위해서는 우선적으로 필요한 데이터가 메모리에 적재되어 있는지를 우선적으로 확인한다. 만약 데이터가 적재 되어 있다면 바로 화면에 디스플레이하고 메모리에 존재하지 않으면 해당 데이터를 찾아 메모리에 적재 시킨 후 출력한다.



(그림 3) 일반적인 맵 이동

한 맵에서 다른 맵으로 이동을 마친 후, 이동한 맵에서 화면에 보이는 모든 캐릭터들을 출력하기 위해서 클라이언트는 필요한 데이터를 모두 로딩한다. 그림 4 에서 하나의 캐릭터 모델을 출력하기 위한 단계를 설명하고 있다. 해당 캐릭터 모델이 메모리에 로딩되어 있으면 바로 화면에 출력하고 로딩되어 있지 않으면 파일에서 해당 데이터를 찾아 메모리에 로딩 시킨 후 화면에 출력한다.

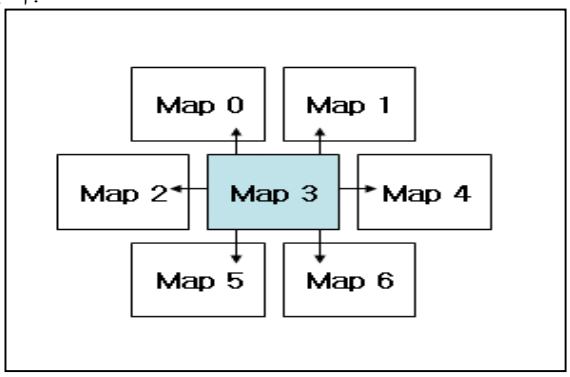


(그림 4) 화면 출력을 위한 단계

3.2 예측 로딩을 통한 방식에서 맵 이동 처리

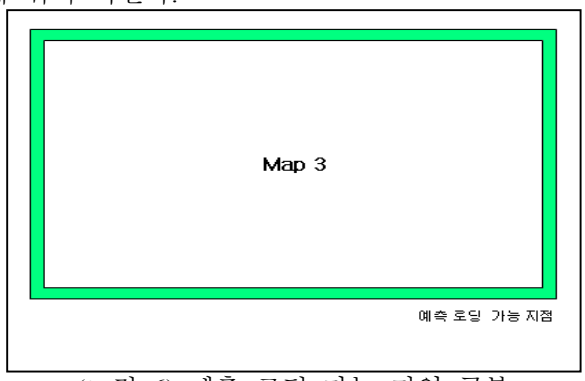
맵 서버들이 격자로 구성된 경우에는 하나의 맵에서

다른 맵으로 이동하는 경우의 수가 많아져서 미리 예측 로딩해야 되는 데이터가 늘어나게 된다. 예측 로딩을 통한 구조에서는 다른 맵으로 이동 할 수 있는 경로를 줄이기 위해 격자가 아닌 hexagon 형태로 변형한다. 그림 5 는 hexagon 형태로 변형된 맵 서버 구조를 보여준다.



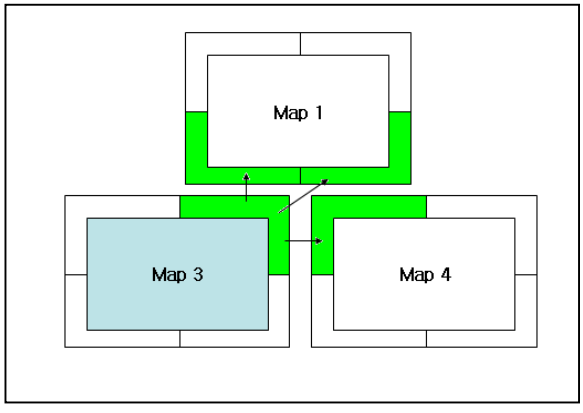
(그림 5) hexagon 로 변형된 맵 구조

각 맵에서의 끝부분에서만 다른 맵으로 연결 되어 이동 할 수 있으므로 예측 로딩이 필요한 지역을 끝단에 위치 시킨다.



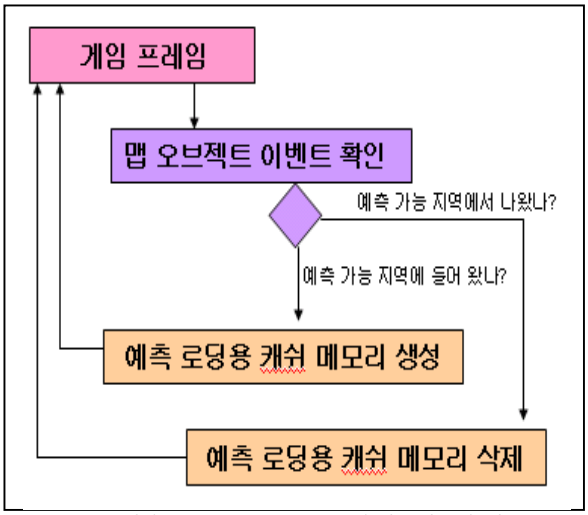
(그림 6) 예측 로딩 가능 지역 구분

하나의 맵의 나눠진 예측 가능 지역에서 다른 맵의 예측 가능 지역으로 이동할 수 있는 경로를 구분해 놓았다. 한 맵의 예측 가능 지역을 4 부분으로 분할하면 다른 맵의 어느 지역으로 이동할 것인지가 예측 가능하다. 그림 7 에서와 같이 하나의 지역에서는 최대 3 군데의 예측 가능 지역이 예측된다.



(그림 7) 다른 지역으로의 이동 예측

해당 맵의 예측 가능 지역으로 들어온 클라이언트는 다른 맵의 예측 지역에 들어 있는 데이터를 미리 캐쉬에 적재 시킨 후 해당 지역에서의 클라이언트가 그 지역을 벗어 났을 경우에는 캐쉬를 삭제한다. 그림 8 에서 게임 프레임이 맵 오브젝트에 대한 이벤트를 확인하는 단계에서 캐릭터가 예측 가능 지역으로 들어오고 나가는 것을 확인하는 단계에서 캐쉬를 생성 및 삭제하는 절차를 보여준다.

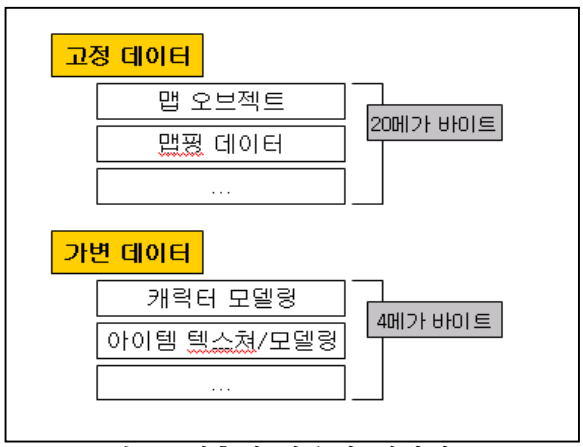


(그림 8) 예측 로딩용 캐쉬 생성 및 삭제

4. 성능분석

4.1 맵 이동 시 필요 데이터

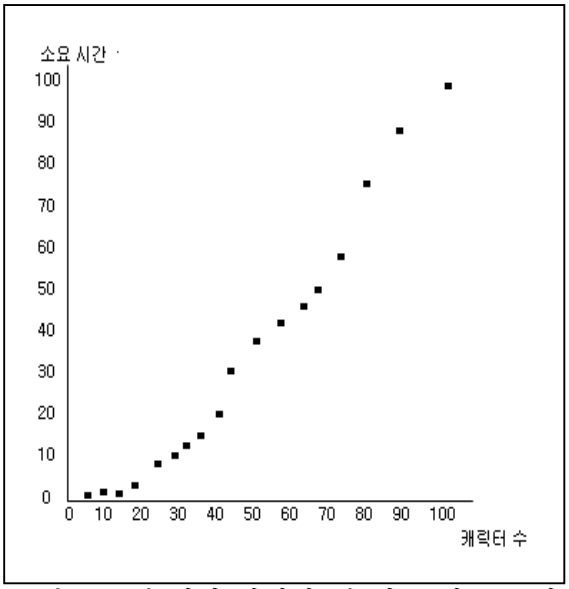
해당 맵에서 다른 맵으로 이동 하였을 경우에 필요로 하는 데이터는 고정적인 것과 가변적인 것으로 나눌 수 있다. 고정적인 데이터는 해당 맵의 데이터와 맵 오브젝트가 있고, 가변적인 데이터로는 캐릭터의 모델링 데이터와 캐릭터가 소유하고 있는 아이템이나 복장 등의 텍스쳐와 캐릭터가 사용하는 이펙트 데이터가 있다. 각 데이터의 크기는 게임의 특성에 따라 크게 달라 질 수 있으므로 객관적인 성능분석을 위해 고정적인 데이터는 20Mbyte 로, 가변적인 데이터는 한 캐릭터 당 4Mbyte 로 크기를 고정시켜 측정하였다. 그림 9 는 성능 분석을 위해 예측에 사용될 데이터의 종류를 고정 데이터와 가변 데이터로 나눠서 설명해 놓았다.



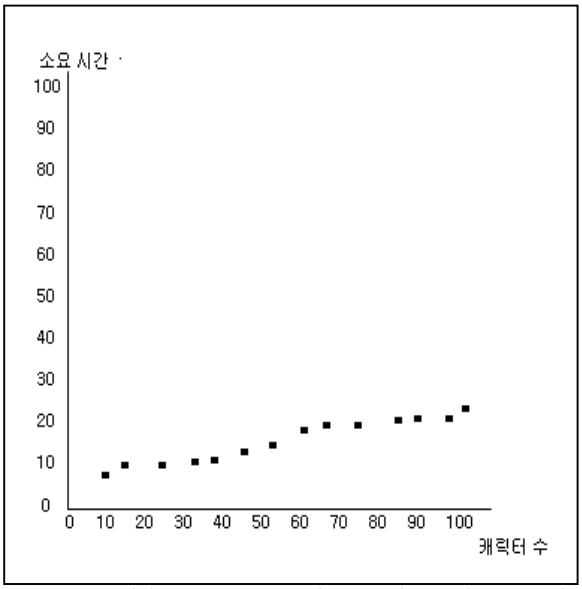
(그림 9) 예측에 사용될 데이터 종류

4.2. 성능분석 결과

이동해야 하는 맵에서 화면에 보여진 캐릭터 수에 따라 메모리를 읽어 들이는 시간이 얼마나 걸리는 지 측정한다[6]. 측정결과의 객관적인 판단을 위하여, 동일한 클라이언트에서 두 방식을 테스트 하였으며, 서버 또한 동일한 서버를 사용하였다. 화면 출력을 위한 라이브러리는 DirectX9.0b 라이브러리의 렌더링 함수를 사용하였고, 객관적인 테스트 측정을 위해 파일을 읽어 들이기 위한 함수는 ::fopen()을 사용하고 메모리를 저장하기 위한 함수는 ::memcpy()를 사용하였고 소요시간은 ms 로 측정하였다.



(그림 10) 맵 서버 방식의 맵 이동 시 소요시간



(그림 11) 예측 로딩을 통한 맵 이동 시 소요시간

그림 10 과 그림 11 에서 각 방식에 따른 성능 측정 결과를 보여준다. 각 점들은 측정 평균치에서 크게 벗어나지 않는 결과를 나타낸다. 예측 로딩을 사용한 방식은 화면에 출력해야 하는 캐릭터의 수에 상관없이 일정한 시간이 소요되며 기존의 방식은 캐릭터의 수가 많아 질수록 대기 시간이 많이 소요된다.

5. 결론

많은 캐릭터가 동시에 한 지역에 밀집하고 있을 경우 예측 로딩 기법을 통하여 많은 부하를 줄일 수 있다. 하지만 캐릭터의 수가 30 명 이하인 경우에는 일반적인 맵 서버의 방식과 큰 차이가 없다는 것을 성능 분석의 결과가 나타내고 있다. 대규모의 캐릭터의 참가로 진행되는 MMORPG 의 경우, 접속자 수가 30 명 이하인 경우는 드물지만, 캐릭터의 참가가 드문 특정 지역이나 접속자 수를 제한하는 지역에서는 예측 로딩을 통한 부하 감소 기법이 효과를 발휘하기가 어렵다. 적절한 캐릭터의 수에 따라 유동적인 부하 예측 기법을 적절하게 도입하는 기법이 추후 연구 과제로 남아 있다.

참고문헌

[1] 전재우, "대용량 롤-플레잉 게임을 위한 규모조절 능력이 있는 고성능 게임 서버 구조", 工業技術硏究所論文集, Vol.21, 2002.

[2] Johannes Färber, "Network game traffic modelling", Proceedings of the 1st workshop on Network and system support for games, April 2002.

[3] Cho, Namjae Baek, Seung Ik Ryu, Kyoungmun, "An Exploratory Investigation Of Player Loyalty To Online Games", 韓國經營科學會誌 , Vol.26 No.2 2001.

[4] Tom Jehaes, "Access Network delay in Networked Games", Proceedings of the 2nd workshop on Network and system support for games, May 2003.

[5] 유현우, "LAGS : 온라인 게임 시스템을 위한 계층 구조", 고려대학교 공학대학원 석사 논문집, 2003.

[6] 정종민, "실시간 온라인 네트워크 게임 환경에서의 게임 트래픽 특성 분석", 産業技術硏究, Vol.20 No.2, 2000.