2Layer-Virtual Cell 방식을 이용한 분산 MMORPG게임서버[1)]

MMORPG Distributed Game Server using 2Layer-Virtual Cell

장수민, 박용훈, 유재수
충북대학교

Jang Su-Min, Park Yong-Hoon, Yoo Jae-Soo
Chungbuk National Univ

요약

다수 사용자용 온라인 게임은 온라인 서비스들 중에 중요한 부분을 차지하고 있다. 최근에는 네트워크를 통한 온라인 서비스를 이용하는 사용자들의 증가로 인해 서버에 부하가 가중되고 있다. 본 논문에서는 이와 같은 문제를 해결하는 2Layer-Virtual Cell방식을 이용한 분산 MMORPG(Massively Multi-player Online Role Playing Game)게임 서버를 제안한다. 제안하는 방식은 많은 사용자들을 위한 MMORPG 분산 게임서버에 효과적인 해결책을 제공한다.

Abstract

An important application domain for online services is an interactive, multiplayer game. in recent, many increase of users that use on-line services through networks have caused a heavy load to the server. In this paper, we propose a MMORPG distributed game server using 2Layer-Virtual Cell. Our method provides more effective solution of MMORPG distributed game server for large numbers of users.

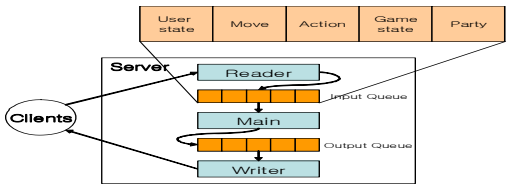
I. 서 론

최근 MMORPG 게임 서버는 이전 보다 넓은 가상의 다양한 공간에서, 보다 많은 사용자를 처리해야 하는 MMORPG 게임으로 발전 되어 가면서 MMORPG 게임 서버에 대한 부하를 가중 시키고 있다. 본 논문에서는 MMORPG 게임에서 가장 많이 사용하는 기존 셀방식이 갖고 있는 문제점을 제시하고 이러한 문제점을 해결하는 2Layer-Virtual Cell방식을 제안한다. 본 논문에서 제시하는 2Layer-Virtual Cell방식은 분산처리에 적합하도록 설계하였다. MMORPG 게임서버의 부하를 낮추는 방법으로 N_Server 분산서버를 만들 때 고려해야 할 여러 가지 변수를 제시하고 최적화하는 방법을 제안한다.

II. 관련연구

1. MMORPG게임서버의 패킷처리

MMORPG 게임서버에서 다수의 클라이언트가 서버에 패킷들을 보내면 서버의 패킷 Reader는 클라이언트가 보낸 정보를 입력큐에 쌓는다. 그런 다음 서버는 입력큐에서 클라이언트가 보낸 패킷들을 꺼내어 각각의 패킷에 맞는 일을 처리하여 그 결과를 출력큐에 쌓는다. 이렇게 쌓인 출력큐는 패킷 Writer를 통하여 클라이언트에게 보낸다. 여기서 클라이언트가 서버와 서로 주고 받는 주된 패킷은 그림1 과 같이 참여자의 상태변화 패킷과 이동패킷, 행동패킷, 게임상태변화패킷, 파티상태패킷 등이 있다[1]. 이동패킷과 행동패킷은 각각 나누어져 보내지는 경우보다 같이 발생할 경우가 더 많다. 여기서 중요한 점은 이러한 패킷 중에 이동패킷과 행동패킷이 거의 서버와 클라이언트가 주고받는 패킷의 대부분을 차지한다. 이처럼 이동패킷과 행동패킷은 셀방식으로 처리하는 서버에서는 주변의 셀정보를 매우 빈번히 참조한다는 것을 의미한다.



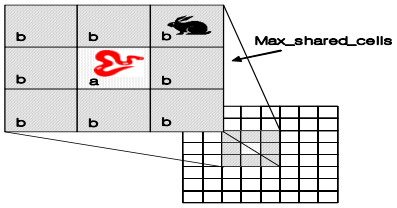
▶▶ 그림 1. MMORPG게임서버의 주된 패킷처리

2. 셀 기반의 MMORPG게임서버

셀기반의 MMORPG 게임서버는 게임플레이어의 활동영역을 일정한 크기 셀로 나누어 위치정보 및 속성정보를 유지하는데 주로 사용하고 있다. 이러한 셀기반의 게임서버는 아주 심플하면서 이벤트가 발생했을 경우 관련된 오브젝트에게 이벤트를 알려주기 위해 주변 오브젝트관리 및 접근이 용이해야 하는데, 셀기반의 방식이 적합하다. 셀방식 이외의 Quad-Tree,

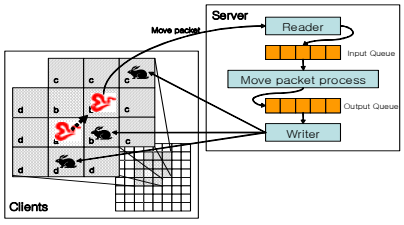
* 본 연구는 한국과학재단 목적기초연구(특정기초연구 과제번호 R01-2003-000-10627-0)지원 및 산업자원부의 지역혁신 인력양성사업의 연구결과로 수행되었습니다.

R-Tree 등 다양한 방법이 있지만 이러한 방식은 MMORPG 게임의 특성상 오브젝트들의 빈번히 노드 이동과 정보 갱신으로 인하여 검색비용 및 유지비용이 많아서 사용하기 적합하지 않다. 셀 기반의 MMORPG게임서버에서 셀 크기는 다양한 크기로 나누어 사용한다. 그러나 보통 셀 의 최소크기는 클라이언트의 게임 화면에서 나타나는 가시범위로 정하여 사용한다. 클라이언트가 움직이는 캐릭터의 가시범위를 셀 최소단위로 사용하는 이유는 게임화면에서 클라이언트에게 보여주기 위한 최소단위이다. 클라이언트의 게임화면을 구성하기 위해서 가시범위크기의 셀 하나만 필요하다. 하지만 캐릭터의 위치가 중앙이 아닌 한쪽으로 치우쳐져 있거나 캐릭터가 이동할 경우를 고려하여 주변의 셀 정보까지 필요로 한다. 이러한 이유 때문에 일반적으로 MMORPG 게임서버는 케릭터가 속한 셀에서 한 칸씩 더 확장시킨 셀까지의 정보들을 클라이언트에게 보내게 된다. 그림 2의 셀(a)에 속한 뱀은 서버로부터 셀(a)의 정보뿐만 아니라 셀(b)들의 토끼 정보까지 요청하여 처리한다. 이러한 처리과정에는 중요한 규칙이 있는데, 셀(a)에 속한 게임 플레이어들은 최대 9개의 셀 정보(Max_Shared_Cells)와 관련이 있고 나머지 셀하고는 독립적이라는 것이다.



▶▶ 그림 2. 셀기반에서 공유데이터처리

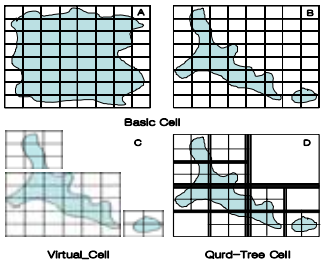
그림3 은 셀(a)에 있는 뱀이 셀(b)로 이동하는 과정이다. 이때 이동패킷을 발생하여 서버에 요청하게 된다. 이동패킷이 서버에 전달되면 서버는 일단 이동패킷을 보낸 뱀의 셀 위치정보를 수정한다. 이때 6개의 셀(c)에 해당하는 토끼들에게는 뱀이라는 오브젝트의 속성과 위치정보를 모두 보낸다. 그리고 셀(a)와 3개의 셀(b)에 있는 토끼들에게는 뱀의 위치 변화정보만을 보낸다. 6개의 셀(d)에 있는 토끼들에게 뱀이 사라졌다는 정보를 보내게 된다. 게임 서버는 주변의 오브젝트 연관성이 많아 근접해 있는 오브젝트의 정보에 대한 접근성이 용이해야 한다. 이러한 접근성에는 셀방식이 적합하다.



▶▶ 그림 3. Move packet처리

3. 기존의 셀기반 MMORPG게임서버의 문제점

그림4 은 8X8 셀로 두 가지 게임에 사용되는 맵을 표현하였다. Array로 표현하면 A[8][8]로 표현된다. A경우는 맵의 모양이 거의 셀을 다 차지하여 자료낭비가 없는 반면 B같은 경우에는 불필요한 셀이 있어 메모리 낭비가 심하다. 이러한 문제점을 없애기 위해 셀을 다양한 크기로 분리하여 사용하는 가상셀방식과 Qurd-Tree를 이용한 셀방식이 있다. 그러나 가상셀방식은 특정 셀의 정보를 얻기 위해서는 분리된 가상 셀들 중에 어떤 셀에 속하는지 판단한 후에 접근해야 하는 비용과 다양한 가상 셀을 유지하는 비용이 발생하여 적합하지 않다. 또한 Qurd-Tree를 이용한 셀기반의 방식은 특정 위치의 셀 정보를 찾기 위해서 Qurd-Tree의 Root를 통한 Index Tree를 검색해야 하는 비용이 발생하여 문제점을 가지고 있다. 이처럼 셀기반의 MMORPG게임에서는 셀을 검색하는 비용이 적게 들면서 메모리 낭비가 적도록 구성해야 한다.



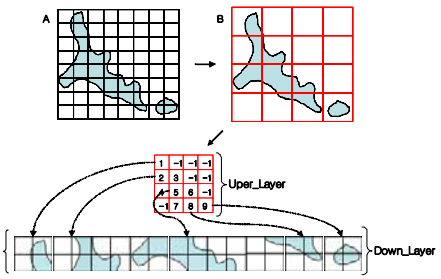
▶▶ 그림 4. 다양한 셀방식

III. 2Layer-Virtual Cell

기존 셀방식에서 필요 없는 셀의 자원낭비를 최소화 하면서 셀의 접근속도가 빠른 방법으로 2Layer-Virtual Cell방식을 제안한다.

1. 2Layer-Virtual Cell 방식

2Layer-Virtual Cell 방식은 Up_Layer와 Down_Layer로 구성 되어 있다. 그림5 에서 보이는 A의 셀기반의 지도구성을 B와 같이 일정크기로 분할하는 Up_Layer가 있고, 실 데이터의 셀로 구성된 Array들이 있는 Down_Layer가 있다. Up_Layer는 지도가 없는 부분은 -1로 표시하구 지도가 있는 부분에는 Down_Layer의 Array의 첨자를 표시한다. 지도가 없어 필요가 없는 부분은 Up_Layer에서 제거하는 것이다. 접근성에서는 Up_Layer가 Down_Layer의 Array의 첨자를 가지고 있기 때문에 수치적 계산방법을 통해서 바로 접근이 가능하다.



▶▶ 그림 5. 2Layer-Virtual Cell방식

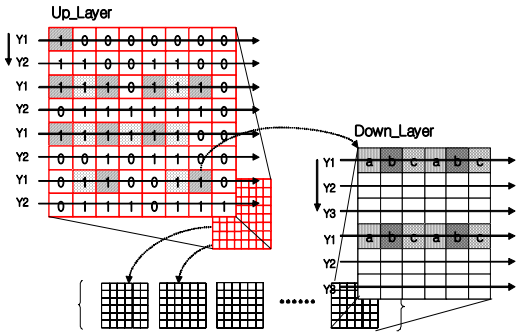
2Layer 방식보다 많은 Layer을 두는 Multi Layer 셀방식을 사용할 수도 있으나 MMORPG게임에서는 주변 셀의 정보를 매우 빈번히 요구하는 연산이 대부분을 차지하고 있어 다차원은 적합하지 않다. 2Layer-Virtual Cell방식을 C언어의 자료구조로 표현하면 아래와 같다. 실 데이터가 저장되는 Struct Real_Cell은 각각의 셀에는 해당되는 User, Monster, Msg등의 정보를 포함한다. Struct Down_Layer는 Struct Real_Cell을 Split_CSize * Split_CSize로 구성된다. Struct Down_Layer를 Array로 갖는 D_Arr는 지도가 표시되어 유효한 수(Real_DCount)만큼 선언한다. Int Up_Layer는 Total_CSize/Split_CSize * Total_CSize/Split_CSize 만큼 Array로 선언하여 지도가 없으면 -1로 표현하구 지도가 있다면 D_Arr의 첨자를 저장한다.

```
Struct Real_Cell{
        Char flag;
        Struct Users;
        Struct Monsters;
        Struct Msgs;
};
Struct Down_Layer {
        Struct Real_Cell C_Arr[Split_CSize][ Split_CSize];
};
Struct Down_Layer D_Arr[Real_DCount];
Int    Up_Layer [Total_CSize/Split_CSize][ Total_CSize/Split_CSize];
특정 셀에 해당하는 User접근하는 구문은 아래와 같다.
If( ( t = Up_Layer [Div(x, Split_CSize)][ Div(y, Split_CSize)] ) >= 0 ) {
        User_temp=D_Arr[t].C_Arr[Mod(x, Split_CSize)] [Mod(y,
        Split_CSize)].Users;
}
```

2. 2Layer-Virtual Cell 방식에서 분산처리

2Layer-Virtual Cell에서 분산은 앞에서 언급한 한 셀에 속한 게임플레이어들은 최대 9개의 셀 정보(Max_Shared_Cells)를 제외한 정보하고는 무관하다는 중요한 사실을 기반으로 분산처리 된다.

그림 6에서 Up_Layer에서 지도가 없는 곳은 -1로 표시되지만 편이상 0으로, 지도가 있는 곳은 Down_Layer의 배열첨자로 표현 되지만 1로 표시하였다. Up_Layer에서 분산처리는 셀의 값이 1 이고 X축과 Y축 차이가 각각 한 칸인 셀들에 대한 Thread()를 생성하여 처리한다. 그림에서 ■들은 서로 공유데이터에 대해 독립적이기 때문에 Thread()를 할당되어 분산처리 할 수 있다. 분산처리 순서는 Y1에 해당되는 셀 중에 ■들을 처리하구 처리가 끝나면 □들을 처리한다. 이처럼 Y1열이 끝나면 다음으로 Y2를 처리하여 모든 셀의 일처리를 끝낸다. Up_Layer에서는 실제적인 일처리가 이루어지지는 않는다. Up_Layer가 가리키고 있는 Down_Layer에서 실제적인 일처리가 이루어진다. Down_Layer에서는 Y1열이고 a인 셀들에게 Thread()를 할당하여 분산처리 한다. a가 끝나면 b를 처리하고 마지막으로 c를 처리한다. a, b, c일이 끝나면 Y2열, Y3열 순서로 처리하여 모든 셀에 대한 일처리를 마무리한다. Down_Layer의 셀크기가 3*3 이상이여야 한다는 조건을 만족해야 한다. 그래야 Up_Layer에서 일처리 단위를 Up_Layer의 셀단위로 일처리 할 수 있다.



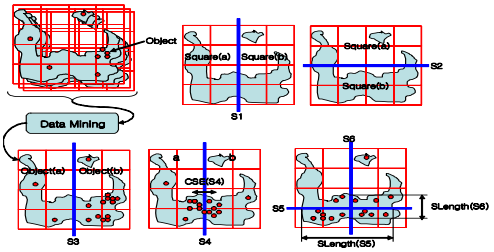
▶▶ 그림 6. 2Layer-Virtual Cell의 분산처리방식

3. N_Server을 이용하는 분산서버

앞에서 언급한 N_Server을 이용하는 분산서버를 구축하기 위해서 2Layer-Virtual Cell방식에서는 Up_Layer을 어떻게 분할 할 것인가가 주된 문제가 된다. N개의 서버로 분할하는 관점에서 고려해야 할 조건들로 지도의 면적, 오브젝트 수, 서버이전 부담률, 경계선의 길이와 같이 네 가지가 있다.

3.1 지도의 면적

그림 7에서 첫 번째 분할 S1과 S2를 보면 각각 세로와 가로로 분할하였는데 분할로 인하여 나누어진 셀에 해당하는 지도의 면적이 서로 다르다. 지도의 면적은 오브젝트들의 이동할 수 있는 공간이 되기 때문에 지도의 면적의 크기가 고려되어야 한다. 분할할 경우 지도의 면적 Square(a)와 Square(b)의 차이 값이 적도록 분할해야 한다.



▶▶ 그림 7. 2Layer-Virtual Cell의 N_Server분할

3.2 오브젝트 수

S3에서처럼 분할된 지도에 Object(a)에는 2개의 오브젝트가 있고 Object(b)에는 13개 있는 형태로 분할이 되었다. 분할할 경우 오브젝트수가 균등하게 나누어지도록 분할하여야 한다. 오브젝트의 수는 분할시점에 오브젝트수를 고려할 수도 있지만 일정한 시간간격으로 데이터를 수집하여 수집된 데이터에서 데이터마이닝을 통하여 표본 데이터를 산출하여 오브젝트 수를 고려하여야 한다.

3.3 서버이전 부담율

S4는 오브젝트 수는 거의 균등하게 잘 분할하였다 그러나 오브젝트들이 분할된 경계선에 몰려있기 때문에 서버에서 서버로 이전할 확률이 높다. 이처럼 서버이전 부담율(Change Server Expense)이 높지 않는 곳을 분할하여야 한다. 서버이전 부담율(CSE(S4))을 구하는 방법은 분할된 경계선 주변 셀의 오브젝트수가 전체 오브젝트 수에 비해 얼마나 많이 있는가를 고려하면 된다.

3.4 경계선의 길이

분할된 경계선의 길이는 서버이전 확률과 밀접한 관계를 가지고 있다. 경계선이 길수록 서버이전 부담율은 높아진다. S5의 SLength(S5)과 S6의 SLength(S6)를 비교하면 SLength(S6)가 경계선의 길이가 짧다. 분할되어지는 경계선은 짧을수록 서버이전 부담율이 적어진다.

이러한 4가지 조건을 바탕으로 서버분할의 최적화 값 Split_Opt(S)은 아래의 식으로 나타낼 수 있다.

$$Split\_Opt(S) = \frac{1}{i*|Square(S(A))-Square(S(B))|+j*|Object(S(A))-Object(S(B))|+k*CSE(S)+l*Slength(S)} \times 100$$

IV. 결 론

본 논문은 기존의 셀기반 MMORPG게임서버의 문제점인 불필요한 셀에서 생기는 메모리 낭비와 가상 셀방식과 Qurd-Tree를 이용한 높은 검색비용과 유지비용의 단점을 보안한 2Layer-Virtual Cell방식을 제안하였다. 2Layer-Virtual Cell방식은 메모리 낭비의 문제점을 해결하구 셀방식에서 Max_Shared_Cells 라는 중요한 사실을 기반으로 분산처리를 위한 Thread들 간에 간섭이 없이 독립된 영역을 분할하여 분산 처리하는 기법을 제시하였다. 또한 최근에 이슈가 되고 있는 N_Server를 이용한 분산처리에 필요한 영역분할의 조건들을 제시 하구 조건들을 이용한 식을 통하여 N_Server로 분할의 최적화할 수 있는 방법을 제시하였다. 향후 고려되어야 할 사항은 실제 게임 데이터 맵을 2Layer-Virtual Cell방식에 적용하여 보고, 실제적인 사용자의 위치정보에 대한 데이터 구축하여 데이터마이닝을 통한 N_Server로 분할하여 그 결과를 피드백하는 연구가 이루어져야 할 것이다.

▌참 고 문 헌▐

[1] Seokjong Yu, "Distributed Game Server and Spatial Partition Technique", Communications of the Korea Information Science Society, Vol.23, No.6, pp.29-35, June 2005.

[2] Ta Nguyen Binh Duong, Suiping Zhou, "A dynamic load sharing algorithm for massively multiplayer online games" IEEE 2003.

[3] Ahmed Abdelkhalek, Angelos Bilas, "Parallelization and performance of interactive multiplayer game servers" IEEE 2004.

[4] Gao Huang,Meng Ye, Long Cheng, "Modeling system performance in MMORPG" IEEE Communications Society Globecom 2004 Workshops

[5] 손민규, 성영락, 오하령, 안현식, 김정윤 "대용량 온라인 게임 서버를 위한 부하 평형 기법", 대한전자공학회 하계종합학술대회 제26권 제 1호 2003.

[6] 이정진, 두길수, 안동언, 정성종, "MMORPG 부하 분산을 위한 동적 맵 분할 시스템 설계" 한국컴퓨터종합학술대회 2005 논문집