효율적인 MMORPG 분산 게임서버

An Efficient MMORPG Distributed Game Server

장수민, 유재수
충북대학교 정보통신공학과

Su-Min Jang(jsm@netdb.chungbuk.ac.kr), Jae-Soo Yoo(yjs@chungbuk.ac.kr)

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-Cell방식을 이용한 분산 MMORPG(Massively Multi-player Online Role Playing Game)게임 서버를 제안한다. 제안하는 방식은 많은 사용자들을 위한 MMORPG 분산 게임서버에 적합한 해결책을 제공한다. 성능평가는 제안하는 MMORPG 분산게임서버가 기존방법에 비해 메모리 사용량과 처리속도에서 성능이 우수함을 보인다.

■ 중심어 : | 분산서버 | 게임 | 통신시스템 트래픽 | 분할알고리즘 |

Abstract

An important application domain for online services is an interactive, multi-player 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(Massively Multi-player Online Role Playing Game) distributed game server using 2Layer-Cell. Our method provides efficient solution of a MMORPG distributed game server for large numbers of users. It is shown through the experiments that our method outperforms existing methods in terms of memory utilization rate and processing speed.

■ keyword : | Distributed Server | Games | Communication System Traffic | Partitioning Algorithm |

I. 서 론

최근 온라인 서비스 중 온라인게임의 비중이 크게 높아지고 있다. 그중에 MMORPG게임 서버는 이전 보다 많은 사용자를 처리해야 하는 MMORPG게임으로 발전되어 가면서 MMORPG게임 서버에 대한 부하를 가중시키고 있다. MMORPG게임에서 많이 사용하는 기존 방식들[8](Cell-Based, Quad-Tree, R-Tree)은 많은 문제점을 가지고 있다. 이러한 문제점으로는 가상공간을 표현하는데 있어 불필요한 메모리낭비와 동시성 제어에 필요한 공유데이터의 느린 처리속도 등이 있다. 본 논문에서는 이러한 문제점을 해결하기 위하여 2Layer-Cell방식을 제안한다. 본 논문에서 제시하는 2Layer-Cell방식은 Upper_Layer와 Down_Layer로 구성하여 메모리의 낭비를 최소화하고 동시성 제어를 위한 공유데이터에 대한 처리속도를 극대화 하였다. 또한 다수의 클라이언트를 효과적으로 처리하기 위한 분산처리에 적합하도록 설계하였다. MMORPG게임서버의 부하를 낮추는 방법으

* 본 연구는 2006년도 충북대학교 학술연구지원사업의 연구비 지원에 의하여 수행되었습니다.

접수번호 : #061120-005
접수일자 : 2006년 11월 20일
심사완료일 : 2007년 01월 09일
교신저자 : 유재수, e-mail : yjs@chungbuk.ac.kr

로 분산서버방식을 적용하였다. 어떻게 서버의 부하를 분산 할 것인가에서 고려해야할 때, 여러 가지 중요한 선택요소가 되는 것을 본 논문에서 제시하고 최적화하는 방법을 제안한다. 또한 기존의 방식[8]과 본 논문이 제안하는 2Layer-Cell방식을 성능평가 및 분석을 하여 기존의 방식보다 메모리사용량과 처리 속도에서 탁월한 결과를 제시하였다.

본 논문의 구성은 먼저 2장에서 관련연구에 대해서 알아보고, 3장에서는 제안하는 2Layer-Cell방식과 분산서버를 나누어 설명한다. 4장에서는 성능평가 및 분석에 대한 설명을 하고, 마지막으로 5장에서는 결론을 맺고 향후연구방향을 제시한다.

II. 관련연구

1. MMORPG게임서버의 패킷처리

MMORPG 게임서버는 다수의 클라이언트와 패킷을 주고 받으면서 게임을 진행하게 된다. 이러한 처리과정을 보면 다음과 같다. MMORPG 게임서버에서 다수의 클라이언트가 서버에 패킷들을 보내면 서버의 패킷리더(Reader)는 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 라이터(Writer)를 통하여 클라이언트에게 보낸다[2]. 여기서 클라이언트가 서버와 서로 주고받는 여러 가지 형태의 패킷이 생기는데 이러한 패킷을 분류하면 [그림 1]과 같이 참여자의 상태변화패킷(참여자의 로그아웃, 로그인)과 이동패킷(참여자의 위치이동), 행동패킷(조작, 공격명령, 스킬사용, 채팅), 게임상태변화패킷(레벨업, 아이템습득, 아이템장착/해체), 파티상태패킷(파티설정, 파티해제)등이 있다[1]. 이러한 패킷들을 분석하여 보면 이동패킷과 행동패킷은 각각 나누어져 보내지는 경우보다 같이 발생할 경우가 더 많다.

여기서 서버와 클라이언트간의 부하에 영향을 줄 수 있는 중요한 점은 이러한 패킷 중에 이동패킷과 행동패킷이 거의 서버와 클라이언트가 주고받는 패킷의 대부분을 차지한다는 것이다. 이동패킷과 행동패킷은 셀방식으로 처리하는 서버에서는 주변의 셀정보를 매우 빈번히 참조한다는 것을 의미한다. 그래서 서버를 설계할 때 특정위치를 셀에 대한 정보를 얻기 위해 접근하는 시간이 적도록 설계되어야 한다.

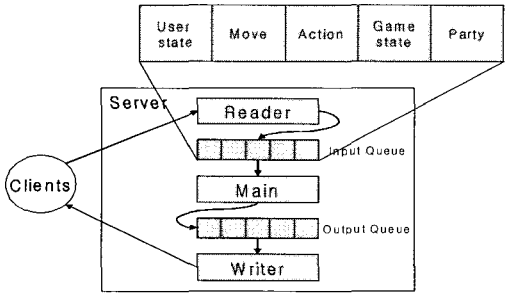


그림 1. MMORPG게임서버의 주된 패킷처리

2. 셀 기반의 MMORPG게임서버

셀기반의 MMORPG게임서버는 게임플레이어의 활동영역을 일정한 크기 셀로 나누어 위치정보 및 속성정보를 유지하는데 주로 사용한다[4]. 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, R-Tree 등 다양한 방법이 있지만 이러한 방식은 MMORPG게임의 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 케릭터의 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.

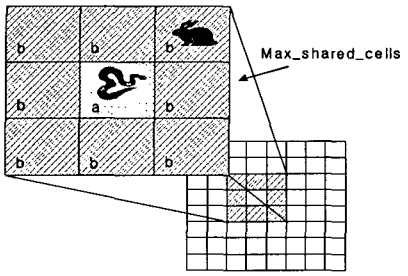


그림 2. 셀기반에서 공유데이터처리

이러한 이유 때문에 일반적으로 MMORPG 게임서버는 케릭터가 속한 셀에서 한 칸씩 더 확장시킨 셀까지의 정보들을 클라이언트에게 보내게 된다.

[그림 2]의 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 게임플레이어들은 최대 9개의 셀 정보(Max_Shared_Cells)와 데이터를 공유하고, 그 이외의 셀하고는 공유되는 데이터부분이 없다. 이러한 규칙을 이용하여 분산처리를 할 때 공유데이터에 대한 동기화에 따른 락(Lock)을 사용하지 않고 분산 처리할 수 있다.

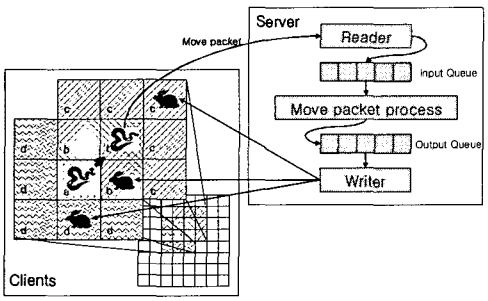


그림 3. 셀기반에서 Move packet처리

[그림 3]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.

3. 기존의 셀기반 MMORPG게임서버의 문제점

셀기반 MMORPG게임서버는 가상의 세계를 표현하기 위해서 가상의 지도를 작은 셀단위로 나누어 사용한다. [그림 4]은 8*8 셀로 두 가지 게임에 사용되는 맵을 표현하였다. 배열로 표현하면 A[8][8]로 표현된다. 이러한 서로 다른 형태의 두가지 지도를 표현한 A와 B가 있다. 이중에 A경우는 맵의 모양이 거의 셀을 다 차지하여 메모리의 낭비가 없다. 그러나 반면 B같은 경우에는 지도가 비워있는 부분이 많아 불필요한 셀이 있는데 이러한 부분은 메모리 낭비가 심하다. 이러한 문제점을 없애기 위해 셀을 다양한 크기로 분리하여 사용하는 가상 셀방식과 Quad-Tree를 이용한 셀방식이 있다. 그러나 가상 셀방식은 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 Quad-Tree를 이용한 셀기반의 방식은 특정 위치의 셀 정보를 찾기 위해서 Quad-Tree의 Root를 통한 Index Tree를 검색해야 하는 비용이 발생하여 문제점을 가지고 있다. 이처럼 셀기반의 MMORPG 게임에서는 셀을 검색하는 비용이 적게 들면서 메모리 낭비가 적도록 구성해야 한다.

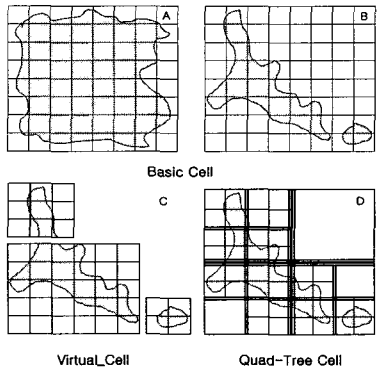


그림 4. 다양한 셀방식

4. 일반적인 온라인 분산 게임서버구조

온라인게임서버는 대부분 분산서버이다. 아무리 한 개의 서버가 능력을 최적화 한다하여도 한 개의 서버는 한계점을 가지고 있다. 그래서 온라인 게임서버는 여러 개의 서버를 사용하여 다양하게 구성한다. [그림 5]는 여러 가지 방식의 구성도이다. A는 하나의 서버가 여러 개의 클라이언트와 접속하는 중앙 집중형의 기본적인 방식으로, 다수의 클라이언트를 처리하기에는 서버에 대한 부하가 많기 때문에 단순한 게임 및 소수의 클라이언트를 처리 하는데 주로 사용된다. B와 같은 분산게임서버구조는 크게 대칭서버(Replicated Server)와 비대칭서버(Non- Replicated Server)로 구분한다. C와 같이 보다 복잡한 구조로는 클라이언트 간에 통신이 병합되어 있는 방식도 있다. 이처럼 다양한 게임서버의 구조는 대부분 더 많은 클라이언트를 분산 처리하여 부하를 적게 가져가기 위한 방법[6]들이다.

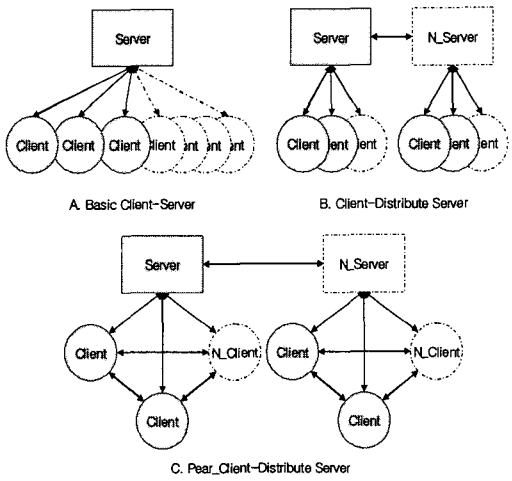


그림 5. 다양한 분산게임서버구조

III. 2Layer-Cell

기존 셀방식에서 발생했던 문제점인 메모리 낭비를 최소화 하면서 Quad-Tree방식에서 이웃하는 셀의 정보에 접근하는 속도 보다 빠른 방법으로 2Layer-Cell방식을 제안한다.

1. 2Layer-Cell 방식

2Layer-Cell 방식은 Upper_Layer와 Down_Layer로 구성 되어 있다. [그림 6]에서 보이는 A의 셀기반의 지도 구성을 B와 같이 일정크기로 분할하는 Upper_Layer가 있고, 실 데이터의 셀로 구성된 Array들이 있는 Down_Layer가 있다. Multi-Leveled-Grid 방식과 Quad-Tree-Cell방식에서 레벨을 2레벨로 제한하는 방식과 비슷하게 보인다. 그러나 이러한 방식과 2Layer-Cell 방식은 전혀 다른 구조이다. 2계층으로 나누어진다하여도 2Layer-Cell 방식의 Upper_Layer은 셀의 정보를 갖는 것이 아니라 아래 계층이 갖고 있는 정보나 Upper_Layer단위에 대한 정보를 갖는 구조이다. Upper_Layer는 지도가 없는 부분은 -1로 표시하고 지도가 있는 부분에는 Down_Layer의 Array의 첨자를 표시한다. 지도가 없어 필요가 없는 부분은 Upper_Layer에서 제거하는 것이다. 접근성에서는 Upper_Layer가 Down_Layer의 Array의 첨자를 가지고 있기 때문에 수치적 계산방법을 통해서 바로 접근이 가능하다.

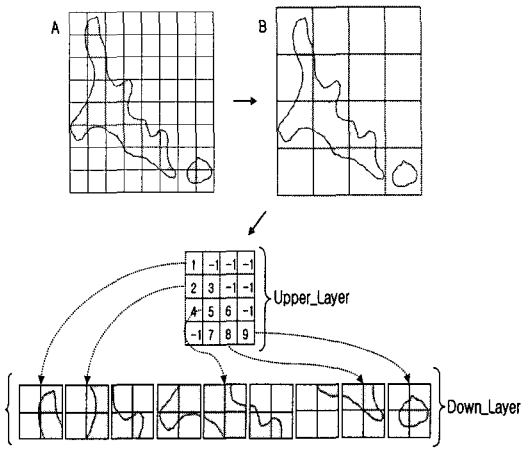


그림 6. 2Layer-Cell방식

2Layer방식보다 많은 Layer를 갖는 Multi Layer 셀방식을 사용할 수도 있으나 MMORPG게임에서는 주변 셀의 정보를 매우 빈번히 요구하는 연산이 대부분을 차지하고 있어 다차원은 적합하지 않다. 2Layer-Cell방식을 C언어의 자료구조로 표현하면 아래와 같다.

```
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          Upper_Layer   [Total_CSize/Split_CSize][
Total_CSize/Split_CSize];
```

실 데이터가 저장되는 Struct Real_Cell은 각각의 셀에 위치하는 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 Upper_Layer는 Total_CSize/Split_CSize * Total_CSize/Split_CSize 만큼 Array로 선언하여 지도가 없으면 -1로 표현하고 지도가 있다면 D_Arr의 첨자를 저장한다. 특정 셀에 해당하는 User접근하는 구문은 아래와 같다.

```
If((t=Upper_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-Cell 방식에서 분산처리

2Layer-Cell방식을 이용한 분산처리는 앞에서 언급한 것과 같이 한 셀에 속한 게임플레이어들은 최대 9개의 셀 정보(Max_Shared_Cells)를 제외한 정보하고는 무관하다는 중요한 사실을 기반으로 분산 처리한다.

[그림 7]에서 Upper_Layer에서 지도가 없는 곳은 -1로 표시되지만 편이상 0으로, 지도가 있는 곳은 Down_Layer의 배열첨자로 표현 되지만 1로 표시하였다. Upper_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[5]. 분산처리 순서는 Y1에 해당되는 셀 중에 ▨들을 처리하고 처리가 끝나면 ▩들을 처리한다. 이처럼 Y1열이 끝나면 다음으로 Y2를 처리하여 모든 셀의 일처리를 끝낸다. Upper_Layer에서는 실제적인 일처리가 이루어지지는 않는다.

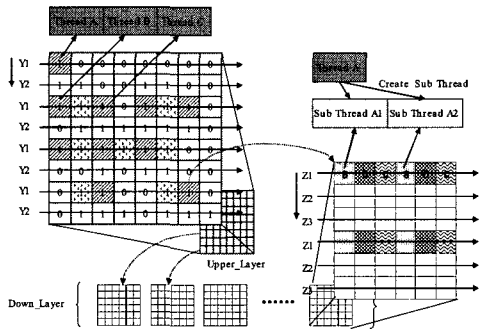


그림 7. 2Layer-Cell의 분산처리방식

Upper_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 셀 크기가 3*3 이상이여야 한다는 조건을 만족해야 한다. 그래야 Upper_Layer에서 일처리 단위를 Upper_Layer의 셀단위로 일처리 할 수 있다.

3. 분산서버를 위한 분할 조건

분산서버를 구축하기 위해서 2Layer-Cell방식에서는 Upper_Layer을 어떻게 분할 할 것인가가 주된 문제가 된다[3][7]. 여러 개의 서버로 분할하는 관점에서 고려해야 할 조건들로 지도의 면적, 오브젝트 수, 서버이전 부담률, 경계선의 길이와 같이 네 가지가 있다.

3.1 지도의 면적

[그림 8]에서 첫 번째 분할 S1과 S2를 보면 각각 세로와 가로로 분할하였는데 분할로 인하여 나누어진 셀에 해당하는 지도의 면적이 서로 다르다. 지도의 면적은 오

브젝트들의 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 10]에서는 S2분할한 것보다 S1으로 분할한 것이 보다 적절하게 면적을 나누어져 있다.

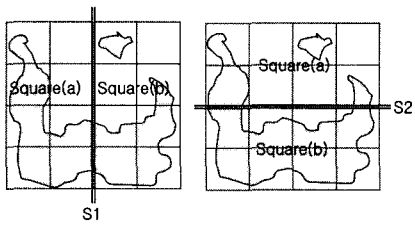


그림 8. 면적의 비교

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 오브젝트 수를 고려할 수도 있지만 일정한 시간간격으로 데이터를 수집하여 수집된 데이터에서 데이터마이닝을 통하여 표본 데이터를 산출하여 오브젝트수를 고려하여야 한다.

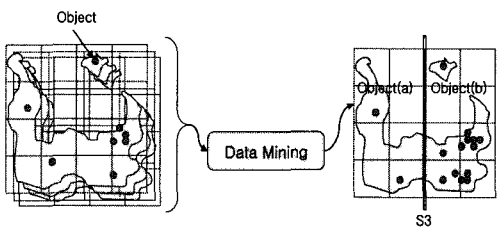


그림 9. 오브젝트수 비교

3.3 서버이전 부담률

S4는 오브젝트 수는 거의 균등하게 잘 분할하였다 그러나 오브젝트들이 분할된 경계선에 몰려있기 때문에 서버에서 서버로 이전할 확률이 높다. 이처럼 서버이전 부담률(Change Server Cost)이 높지 않는 곳을 분할하여야 한다.

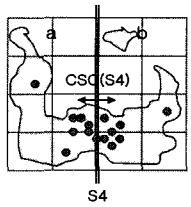


그림 10. 서버이전비용비교

서버이전 부담률(CSC(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 부담률이 적어진다.

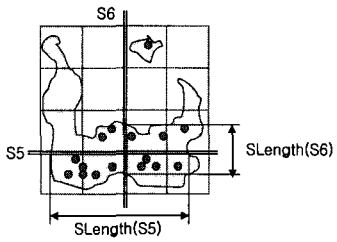


그림 11. 경계선 길이 비교

3.5 분할의 최적화

이러한 지도의 면적, 오브젝트 수, 서버이전 부담률, 경계선의 길이와 같은 네 가지 조건을 바탕으로 서버분할의 최적화 값 Split_Opt(S)은 아래의 식(1)으로 나타낼 수 있다. 분할의 최적화를 고려하여 서버를 분할하여 분산서버를 구축한다.

$$Split\_Opt(S) = \frac{1}{\begin{array}{l} k^* \mid Square(S(A)) - Square(S(B)) \mid + \\ l^* \mid Object(S(A)) - Object(S(B)) \mid + \\ m^* CSC(S) + n^* Slength(S) \end{array}} * 100 \quad (1)$$

IV. 성능평가 및 분석

제안하는 2Layer-Cell방식은 다수 사용자용 온라인 게임을 고려하여 설계하고 구현하였다. 구현한 2Layer-Cell방식의 성능평가하기 위하여 다양한 맵크기와 동시 접속자수를 사용하여 성능을 비교, 분석하였다.

1. 실험 환경

제안하는 2Layer-Cell방식은 기존의 셀 방식, Quad-tree셀 방식과 비교하여 성능을 평가하였다. 실험환경은 펜티엄 IV 2.0GHz에 512Mbyte의 메모리을 가진 컴퓨터에 LINUX환경에서 측정하였다. 비교측면에서 다양한 크기의 맵과 동시접속자 수를 변화하여 성능평가 하였다.

2. 메모리사용량

기존 셀방식과 2Layer-Cell방식을 비교대상으로 하여 메모리 사용량을 비교하였다. 지도의 크기를 1024*1024개의 셀과 2048*2048, 4096*4096개의 셀로 구분하여 비교하였다. [그림 12]와 같이 기존 셀방식은 불필요한 곳까지 메모리를 차지하기 때문에 2Layer-Cell방식에 비하여 다소 높게 차지하였다. 2Layer-Cell방식은 불필요한 메모리 낭비를 제거하기 위하여 Upper_Layer를 이용하였다. 그러나 메모리의 량은 Quad-tree를 이용한 방식과 거의 비슷하였다.

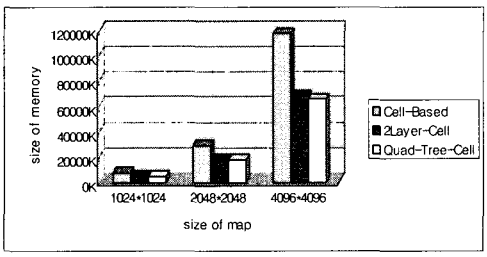
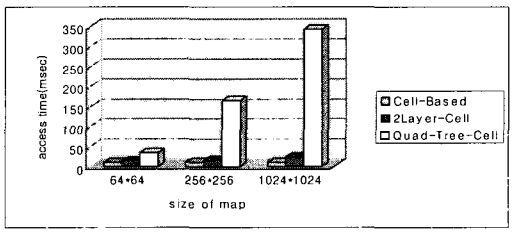


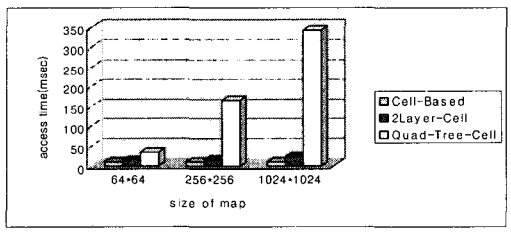
그림 12. 메모리 사용량

3. 접근시간

검색속도는 다양한 맵크기를 사용하여 Quad-tree와 비교 하였다. Quad-tree는 맵의 크기가 커질수록 index의 차원수가 높아지면서 접근시간이 늘어났다. 그러나 2Layer-Cell방식의 경우에는 맵의 크기에 상관없이 접근시간이 거의 일정하였다. [그림 13]과 같이Quad-tree는 맵크기 64*64의 경우와 1024*1024 경우를 비교해보면 접근시간이 10배정도 늘어난 것을 볼 수 있듯이 맵이 크기가 커질수록 검색비용이 많다는 것을 알 수 있다. 기존 셀방식과 비교에서는 접근시간이 약간 더 차이가 있지만 거의 비슷하다. 동시 접속하는 사용자수를 변화하는 실험에서도 비슷한 결과가 나왔다.



Various size of maps , 1000clients



Various numbers of Clients, 1024*1024 map

그림 13. 접근시간

V. 결 론

본 논문은 기존의 셀기반 게임서버의 문제점인 불필요한 셀에서 생기는 메모리 낭비와 가상 셀방식과 Quad-Tree를 이용한 높은 검색비용과 유지비용의 단점을 보안한 2Layer-Cell방식을 제안하였다. 2Layer-Cell방식은 메모리 낭비의 문제점을 해결하고 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 필

요한 영역분할의 조건들을 제시 하고 조건들을 이용한 식을 통하여 N_Server로 분할의 최적화할 수 있는 방법을 제시하였다. 향후 고려해야 할 사항은 실제 게임 데이터 맵을 2Layer-Cell방식에 적용할 것이다. 또한 게임서버를 N_Server로 분할할 때, 실제적인 사용자의 위치정보 에 대한 데이터마이닝 기법을 적용하고자 한다.

참 고 문 헌

[1] A. Abdelkhalek and A. Bilas, "Parallelization and performance of interactive multiplayer game servers," IPDPS, p.72, 2004.

[2] A. Abdelkhalek, A. Bilas, and A. Moshovos, "Behavior and performance of interactive multi-player game servers," ISPASS, pp.137-146, 2001.

[3] J. C. S. Lui and M. F. Chan, "An efficient partitioning algorithm for distributed virtual environment," IEEE Transactions on systems Parallel and Distributed Systems, Vol.13, pp.193-211, 2002.

[4] G. Huang, M. Ye, and L. Cheng, "Modeling system performance in MMORPG," GlobeCom, pp.512-518, 2004.

[5] D. R. Butenhof, Programming with POSIX Threads, Addison Wesley, 2000.

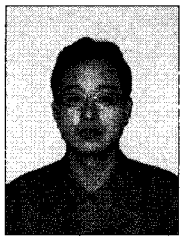
[6] D. Min, E. Choi, D. H. Lee, and B. Park, "A Load Balancing Algorithm for a Distributed Multimedia Game Server Architecture," IEEE International Conference on Multimedia Computing and Systems, pp.882-886, 1999.

[7] N. Baughman and B. Levine, "cheat-proof palyout for centralized and distributed online games," Infocom, pp.104-113, 2001.

[8] Y. Xia and Prabhakar, "SQ+Rtree: efficient indexing for moving object databases," DASFAA, pp.175-182, 2003.

저 자 소 개

장 수 민(Su-Min Jang) 정회원



- 1997년 2월 : 목포대학교 전산통계학과(이학사)
- 1999년 2월 : 충북대학교 정보통신공학과(공학석사)
- 2000년 3월 ~ 현재 : 충북대학교 정보통신공학과 박사과정 중

<관심분야> : 분산처리, 데이터베이스, 게임, 정보검색, 분산 객체 컴퓨터

유 재 수(Jae-Soo Yoo) 종신회원

- 1989년 2월 : 전북대학교 컴퓨터공학과(공학사)
- 1991년 2월 : 한국과학기술원 전산학과 석사(공학석사)
- 1995년 2월 : 한국과학기술원 전산학과 박사(공학석사)
- 1996년 9월 : 충북대학교 전기전자공학부 부교수
- 2006년 4월 ~ 현재 : 충북대학교 전기전자공학부 교수

<관심분야> : 데이터베이스시스템, 멀티미디어 데이터베이스시스템, 정보검색, 분산 객체 컴퓨터