MigAgent를 이용한 Seamless 게임에서의 부하 분산

Load Balancing in Seamless Game with MigAgent

김 법 균* 장 행 진** 유 강 수***
Beob Kyun Kim Hang Jin Jang Kang Soo You

요 약

기존의 다른 어떤 종류의 게임보다도 큰 부하와 거대한 하나의 가상월드가 MMORPG(Massively Multiplayer Online Role-Playing Game)의 중요한 특징이다. 이러한 Seamless 환경에서는, 독립된 게임 서버들이 관리하는 하위의 공간들을 모아 하나의 거대한 가상 월드로 구축한다. 게임 개발자들의 노력에도 불구하고 게이머들은 여전히 클라이언트와 서버, 필드 서버들 간의 상호작용에서 비롯된 과다한 메시지 트래픽에 대해 불만을 가지고 있다. 본 논문에서는 메시지 트래픽을 줄이기 위해 MigAgent를 이용한 새로운 게임 서버 구조를 제안한다. 일반적으로 PC(Player Character)가 다른 필드 서버로 이동할 때 발생하는 메시지 트래픽이 최고조에 이른다. MigAgent는 이런 PC들을 처리하고 예기치 못한 상황에 대처하기 위한 에이전트 역할을 수행한다. 관심영역과 인접 셀의 크기 비율에 따라 그 효과를 분석함으로써 이 시스템의 효과를 분석한다.

Abstract

The load of this kind of games, which is heavier than that of any other precedents, and an enormous seamless virtual world characterize the MMORPG(Massively Multiplayer Online Role-Playing Game) genres. In this seamless environment, gamers can usually accept a set of independent spaces, which is being held by independent game servers, as a single big virtual world. Despite the efforts of game developers, gamers are suffered from huge message traffic which comes from the interaction between client and server and the interaction between field servers. In this paper, new game server architecture using MigAgent is proposed which tries to reduce message traffic. Usually, message traffic reaches the climax when a PC(Player Character) is moving to other field server. MigAgent, designed in this thesis, tries to manage this kind of PCs and to play a role of user's agent to prepare for the unexpected situation. Improvement of this system is shown by the analysis of the effect of the size ratio of AOI (Area of Interest) and AC (Adjacent Cell).

☞ Keyword : 관심 영역, 인접 셀, 그리드, 게임, AOI, Adjacent Cell, Grid, Game

1. 소 개

MMORPG(Massively Multiplayer Online Role-Playing Game)는 수많은 사람들이 동일한 가상공간에서 동시에 상호작용하면서 활동하는 온라인 롤플레잉 게임이다[1-3]. 소수가 모여서 즐기던 고전 RPG롤플레잉 게임과 달리, 인터넷의 발달과 함께 RPG에 머드 게임의 특성을 결합해서 생겨났다. 이 장르의 특성은 기본적으로 스토리가 있는 RPG게임의 특성을 유지하여 게임에 참여하는 모든 플레이어가 동일한 스토리를 즐길 수 있다는 것이다. 게임 내에서는 각 사용자를 대신하는 PC(Player Character)가 그 역할을 대신하게 되며 일종의 아바타(Avatar) 역할을 한다.

MMORPG들은 직면한 문제들을 해결하기 위해 상당한 수준의 자원을 필요로 한다[3]. 예를 들어, 가상공간을 유지해야 하고 높은 수준의 하드웨어가 필요하며 시스템 운영 전담 스태프들이 필요하다. 이러한 이슈들을 해결하기

* 준회원: 전북대학교 BK21-전북지역 전자정보 고급인력 양성사업단 박사후과정
bkyun.kim@gmail.com
** 정회원: 한국과학기술정보연구원 팀장
hjjang@kisti.re.kr
*** 정회원: 전주대학교 교양학부 교수
gsyou@jj.ac.kr
[2006/07/07 투고 - 2006/07/12 심사 - 2006/08/28 심사완료]

위한 개발자들의 노력에도 불구하고 이 장르의 게임들은 여전히 인구과밀 또는 인구부족, 랙(lag), 지원 부족 등을 안고 있다. Peer-to-peer 연결을 바탕으로 구성할 경우 이러한 문제들을 상당부분 해결할 수 있으나 비대칭적인 네트워크 대역폭, 컴퓨팅 능력 부족, 그리고 치팅(cheating)에 대한 취약성 등에서 오는 문제가 더 커진다.

이러한 환경에서는 기존의 한정된 크기의 가상공간을 활용하는 게임과는 비교할 수 없을 정도의 거대한 규모의 가상공간과 많은 수의 게이머들을 유지해야 한다. 이를 위해 대부분의 MMORPG들은 독립된 게임 서버들이 관리하는 작은 단위의 공간을 하위의 지역으로 인식하게 하고 이들을 묶어 거대한 규모의 가상공간을 구축하는 방법을 사용한다. 이렇게 분할된 지역으로 가상공간을 구성하는 방법에도 지역의 분할과 운용 방법에 따라 명시적 지역 분할(Explicit Region Partition) 방식과 묵시적 지역 분할(Implicit Region Partition) 방식이 있다.

명시적 지역 분할 방식은 지역 분할 시 각 서버가 담당하는 지역이 명확하게 확인되는 존(Zone)이라는 단위로 나뉜다. 존 사이의 이동은 계곡이나 다리와 같은 지형적인 특성으로 제한함으로써 각 게임 서버가 PC의 이동에 대해 고려해야 할 경우를 최소화시키며 경우에 따라 PC의 이동시 화면 전환을 통해 서버 사이의 동기화에 필요한 시간을 확보하는 방법을 쓰기도 하며 다른 존에 있는 PC와는 채팅정도만 가능하다. 이러한 방식은 구현이 용이하고 메시지 트래픽을 줄이기 위해 동기화 정보를 압축하여 전송할 수 있다. 전체적으로 메시지 트래픽이 상대적으로 적으며 보다 많은 수의 PC를 수용할 수 있는 장점이 있다.

묵시적 지역 분할 방식은 가상공간을 연속적으로 구성하여 PC로 하여금 각 서버가 담당하는 지역을 자유롭게 이동할 수 있도록 구성한다. 이러한 형태의 가상공간을 운용하는 게임을 Seamless 게임[4]이라 칭하기도 한다. 상대적으로 발생하는 메시지 트래픽이 많고 수용할 수 있는 PC의 수가 적다. 존 사이의 이동이 빈번하게 발생할 수 있으며 처리도 신속하게 이루어져야 한다. 각 영역의 경계에서 활동하는 PC에 대한 처리가 복잡하고 구현이 까다롭다는 단점이 있다. 그러나 구성된 가상공간이 보다 현실에 가까운 형태를 띠고 있으며 게이머가 몰입하기에 더 적합한 형태이기도 하다. 최근의 MMOG의 경우 전체 가상환경의 일부 영역을 Seamless 게임 방식으로 구성하는 사례가 많아지고 있으며 하드웨어의 발전 속도와 게이머들의 욕구 등을 감안하면 Seamless 게임이 대세를 이룰 것으로 보인다.

특히 Seamless 게임 환경의 구현에 있어서 가장 큰 문제점으로 지적되는 메시지 트래픽의 경우, 각 영역의 경계에서 활동하는 PC들의 처리 방법에 따라 그 차이가 더 커진다. 다수의 클라이언트가 동시에 게임을 진행하므로 클라이언트들 사이의 동기화를 위해 클라이언트마다 PC 및 NPC의 상태변화, 기타 객체의 상태변화, 이벤트 지역의 상태변화, 대화 메시지 등과 같은 동기화 정보를 교환해야 하며 게이머들은 영역간의 구분이 없는 것으로 인식하고 활동하면서 다른 게이머들과 상호작용을 할 수 있어야 한다. 따라서 각 클라이언트 머신에서의 물리적인 처리량 계산이 증가하게 되고 렌더링 해야 할 대상(PC 및 객체)의 수가 $O(n)$에 비례하여 증가한다. 또한 서버군 내에서는 특정 영역에 모인 PC의 수가 증가할 경우 영역을 담당하는 서버의 처리량이 $O(n^2)$에 비례하여 증가한다. 예를 들어 특정 영역에 밀집된 PC의 수가 1000이라고 할 경우 동기화 정보의 양은 1,000,000배로 증가하는 것이다[4]. 따라서 이 문제를 해결하는 것이 Seamless 게임 환경을 구축하는데 있어서 가장 먼저 해결해야 할 문제라 할 수 있다[4][5].

본 논문에서는 Seamless 게임 환경에서의 동

기화 메시지 트래픽의 최소화와 이로 인한 체감 성능 저하를 최소화하기 위한 방안으로 MigAgent를 도입한 기법을 제안하고 그 효과를 비교 분석한다. 제 2장에서는 메시지 트래픽을 최소화하기 위한 기존 연구들을 소개하고 제 3장 및 4장에서는 제안하는 시스템의 소개와 실험 및 평가가 이어지며 제 5장에서 결론을 맺는다.

2. 관련 연구

가상환경 내에서 각 게이머들이 인식할 필요가 있는 영역은 일부 지역에 국한되는 경우가 대부분이다. 이러한 영역을 AOI(Area of Interest)이라 하며 관심영역 내 PC들에게만 메시지를 전달함으로써 트래픽의 양을 줄이고자 하는 연구가 진행되어 왔다. 이 기법은 AOI를 정의하는 방법에 따라 클래스 기반, 지역 기반, 격자 기반 방식으로 나뉜다[6-8]. 클래스 기반 방식에서는 갱신될 데이터의 종류에 따라 AOI를 정하며, 지역 기반과 격자 기반 방식에서는 PC의 위치에 따라 갱신될 영역을 정한다. 지역 기반 방식에서는 AOI를 가입 영역과 갱신 영역으로 표시하고, 데이터를 보내는 참가자의 갱신 영역이 받을 참가자의 가입영역과 겹칠 때 데이터를 보낸다. 격자 기반 방식에서는 지역 기반의 방식처럼 갱신 영역과 가입 영역을 사용하지만, 이 영역은 연속적으로 정할 수 있는 영역이 아니라 격자에 의해 미리 나뉜 영역 중에서 선택하는 방식이다[9]. 또한 많은 수의 PC가 좁은 영역에 편중된 경우 메시지의 양을 줄이기 위한 기법도 있다[10]. 이러한 기법들은 각 PC들에게 중요하지 않은 데이터의 불필요한 전송을 최소화하여 네트워크 트래픽과 연산을 줄인다.

PC가 다른 서버가 관리하는 영역으로 이동할 것으로 예측될 경우 이동시 필요한 데이터를 미리 적재하여 부하를 감소시키고자 하는 예측로딩 기법[11]과 맵을 일정 영역으로 분할하여 그룹화한 후 각 그룹을 가상 영역으로 분할하여 데이터를 전송함으로써 서버의 데이터 전송 횟수를 최소화하여 서버의 부하 및 네트워크 트래픽을 최소화하는 기법[12,13]도 있다. 각 프로세스의 종류에 따라 게임 서버를 독립적으로 구축하여 프로세스를 분산시키는 방식[14]의 경우 맵의 확장이 있을 경우 처리가 복잡하며 사용자의 수가 예상치를 초과할 경우 메시지 트래픽의 증가를 감당하지 못해 전체적인 재구성이 필요하다는 단점이 있다.

이와 같이 메시지 트래픽을 최소화하기 위한 기법들은 부분적으로 효과가 있으나 Seamless 게임 환경에서 영역과 영역 사이를 이동하는 PC의 수가 증가할 경우 인접한 게임 서버에서의 PC 정보 등록 및 삭제 등과 같은 부분에서 발생하는 부하와 메시지 트래픽에 대해서는 대책이 없다.

PC가 영역과 영역 사이를 이동할 때 발생하는 트래픽을 최소화하고 객체 검색의 효율성을 높이기 위해 Migration Server라는 별도의 서버를 추가하는 기법이 있다[4]. 각 게임 서버가 관리하는 영역을 Field라 칭하고 이 서버를 필드서버(Field Server)라 한다. 다른 Field로 이동하는 PC는 반드시 Field의 경계면을 거친다는 점에 착안하여 다른 Field와 인접한 경계 부분을 AC(Adjacent Area)라 하며 이 구역 내에 위치한 PC들의 정보를 Migration Server가 관리한다. AC 내에 진입한 PC의 동기화 정보를 Migration Server가 로드하여 인접 필드서버에 등록해 줌으로써 PC의 이동에 의한 필드서버의 지연을 최소화시킨다. 그러나 이 방식의 경우 AC 진입이 빈번한 경우에 대한 대응책은 없으며 인접 Field로 이동하지 않고 AC 내에서만 활동하는 PC들에 대해서도 불필요한 동기화 정보 로딩 작업이 이루어지는 단점이 있다. 실제로 메시지 트래픽 감소 효과는 있으나 PC의 수가 증가 할수록 그 효과는 미미해진다.

실제 상황에서는 트래픽의 폭주를 막기 위해 다수의 채널을 운영하기도 하지만 이런 경우 완전하게 분리된 다수의 가상공간이 운영되는 것이므로 바람직한 방법은 아니다. 또한 과도한 트래픽이 빈번하게 발생할 경우 서버를 증설하기 위해 일시적으로 서비스를 중지하기도 하지만 게이머들이 요구하는 활동의 연속성이 차단된다는 점에서 바람직하지 못하다.

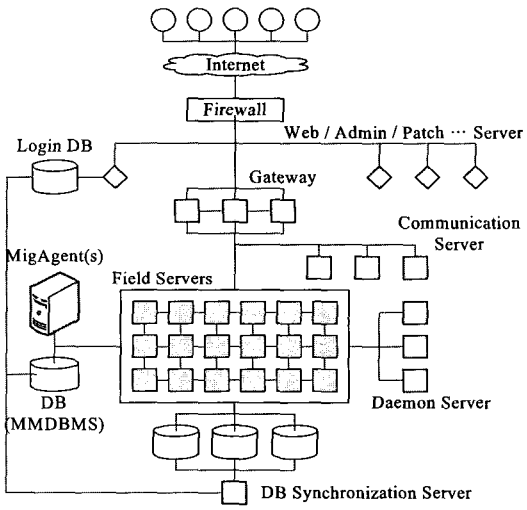
본 논문에서는 Migration Server와 유사한 MigAgent를 도입한 새로운 분산 게임 서버 구조를 제안한다. 인접 Field로 이동하는 PC의 정보를 MigAgent가 관리하며 각 Field 내 AC 진입을 감지하는 과정을 개선함으로써 불필요한 PC 동기화 정보 로딩 작업을 최소화 한다. 또한 MigAgent가 기존의 DBMS가 아닌 MMDBMS (Main Memory DBMS)를 이용하도록 하며 게이머의 대리인(Agent) 기능을 수행할 수 있도록 함으로써 예기치 못한 상황에 대처할 수 있도록 한다. AOI와 AC의 크기에 따라 그 효과를 분석함으로써 설계된 구조의 효과를 분석한다.

3. MigAgent를 이용한 Seamless 게임 설계

3.1 MigAgent를 이용한 Seamless 게임 서버 구조

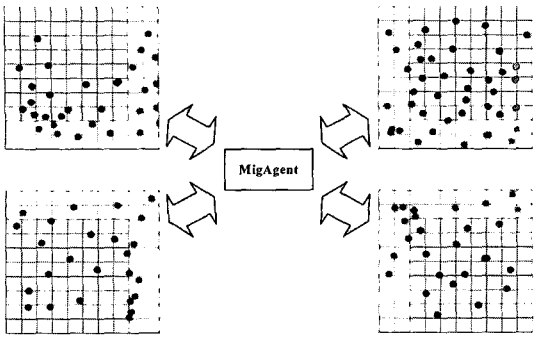
Seamless 게임 서버 구조는 <그림 1>과 같이 묵시적 지역 분할 방식을 이용한 분산형 게임 서버 구조를 따르며 MMDBMS를 이용하는 MigAgent가 추가된다. Login Server는 게이머의 접속에 대한 인증 및 계정 관리를 수행하며 로그인이 이루어질 때 데이터베이스에서 필요한 정보를 찾아 판단한다. 최근 NPC를 별도의 서버에서 제어하도록 함으로써 각 필드서버에서는 PC와 NPC를 거의 구분하지 않고 처리할 수 있도록 함으로써 각 필드서버에서의 부하를 줄이고 개발 비용 및 과정에서의 개선 효과를 얻고자 하는 경우가 많다.

게임 서버를 구성하는 주요 객체들은 다음과 같다.

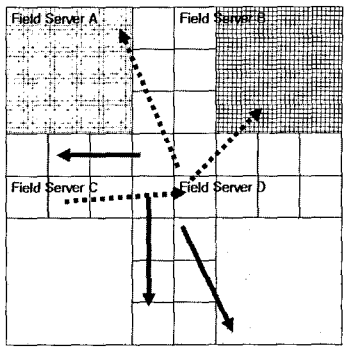


〈그림 1〉 MigAgent를 이용한 Seamless 게임 서버

- PC : 가상공간 내에서 게이머의 아바타 역할을 하는 객체로 게이머의 입력에 따라 방향 변경, 이동 및 각종 액션을 수행한다. 클라이언트에서 그래픽으로 처리되어 렌더링되며 몬스터(Monster)와 같이 게이머의 PC가 아닌 자동화된 PC들을 NPC(Non-PC)라 한다.
- Field와 필드서버(Field Server) : 묵시적 지역 분할 방식은 가상공간을 연속적으로 구성하여 PC로 하여금 각 서버가 담당하는 지역을 자유롭게 이동할 수 있도록 구성된다. 전체 가상공간을 Field라 불리는 작은 단위로 나누어 독립된 서버에 분산 처리하도록 하며 이 서버들을 필드서버라 한다. 각 PC는 마지막 저장 위치를 기준으로 필드서버에 등록되고 재접속 시 이전에 등록되었던 필드서버와 연결되며 각 필드서버는 게임에서의 전반적인 로직들을 처리하는 게임 서버의 역할을 수행한다.



〈그림 2〉 MigAgent의 역할



〈그림 3〉 PC의 이동 특성

- 메시지
  서버와 클라이언트 또는 서버와 서버 사이에 교환되는 정보이다. PC의 이동 방향 및 속도 등에 관련된 정보와 PC가 활동 Field를 바꿀 때 필드서버들이 교환하는 정보 등을 포함하는 것으로 게임 플레이에 관련되어 발생하는 모든 메시지를 통칭한다. 메시지의 양 또는 처리속도는 게이머들에게는 게임 플레이와 관련된 지연시간으로 나타난다.

3.2 MigAgent를 이용한 부하 분산

전체 가상공간 W는 Field들로 구성(식 1)되며 임의의 Field F_i는 Cell이라 불리는 최소 단위로 구성된다(식 2). 각 Cell 내에 존재하는 PC 및 NPC는 Cell이 속한 필드서버에 의해 관리된다. 인접 Field와 경계를 이루는 Cell들을 AC(Adjacent Cell)이라 하며 이 Cell에 진입한 PC들을 일반적으로 다른 Field로 이동할 가능성이 높은 PC들로 판단할 수 있다.

$$W = \{ F_1, F_2, F_3, \cdots, F_m \} \quad (1)$$

$$F_i = \{ C_{i1}, C_{i2}, C_{i3}, \cdots, C_{\in} \} \quad (2)$$

$$AC_i = \{ C_{i1}, C_{i2}, C_{i3}, \cdots, C_{ik} \},\ where \quad k < n \quad (3)$$

본 논문에서는 AC 영역 내에 있는 PC들을 처리하는 MigAgent를 두고 이를 통해 인접 Field로 이동할 가능성이 높은 PC들을 파악하여 이들의 동기화 정보를 관리함으로써 인접 Field로 이동할 때 발생하는 메시지 트래픽을 최소화하고 처리속도를 개선한다. 인접 Field로 이동할 것을 예측되는 PC의 동기화 정보는 MigAgent에 등록되고 인접 Field의 AC를 벗어날 때까지 MigAgent가 이 정보를 유지 및 동기화 작업을 수행한다. <그림 2>는 MigAgent의 역할을 기술한 것으로 음영 처리된 AC영역 내의 PC들은 MigAgent가 관리하게 된다.

각 필드의 경계에 위치한 AC에 PC가 진입하게 되면 MigAgent의 관리 대상이 되며 잠재적으로 인접 필드로 이동할 가능성이 있다. 그러나 그 이동 방향을 분석하면 이동할 가능성이 적은 PC들을 선별할 수 있으며 <그림 3>는 이러한 PC들의 이동방향을 표현한 것이다. 실선으로 표현된 이동들은 맵 경계면과 평행한 것들로 MigAgent가 관여하지 않도록 한다. 또한, PC가 AC 내에 진입한 이후 일정한 시간이 지난 후에 MigAgent가 관리하도록 함으로써 일시적인 AC 진입에 대한 MigAgent의 반응을 줄인다.

3.3 AOI와 AC

게임 운영 측면에서 보면, AOI는 필드 서버

가 PC 하나당 관리해주어야 할 영역을 의미하며 그 크기가 커질수록 송수신되는 트래픽의 양은 많아진다. 본 논문에서는 AC 영역을 정한 후 AC내에 진입한 PC를 MigAgent가 관리함으로써 성능향상을 꾀하고 있으므로 AOI 자체의 영역은 실험의 편의를 위해 격자 기반 형식으로 정의하고자 한다. 격자 기반 형식의 정의에서는 격자 단위의 작은 영역을 미리 설정해 두고 이 단위의 집합으로 AOI를 결정한다. 따라서 AOI 또한 AC와 마찬가지로 Cell단위의 집합으로 표현될 수 있다.

$$AOI_i = \{ C_{i1}, C_{i2}, C_{i3}, \cdots, C_{ik} \},\ where \quad k < n \quad (4)$$

AOI의 크기는 필드 서버가 관리할 영역의 크기를 의미하며 AC의 크기는 MigAgent가 관리할 영역의 크기를 의미하므로 AOI의 크기와 AC의 크기와의 비율에 따라 그 성능의 차이가 현격하게 차이날 것으로 예상된다. AOI의 크기가 AC의 크기보다 상대적으로 클 경우 필드 서버와 MigAgent의 트래픽은 기하급수적으로 증가할 것으로 보이며 AOI의 크기가 AC의 크기보다 상대적으로 작을 경우 트래픽은 현저하게 감소할 것으로 예상된다.

3.4 MMDBMS

최근 알고리즘이나 접근 방법 등의 개선을 통한 성능 개선 뿐 아니라 저장 매체의 개선을 통한 성능향상 또한 관심을 끌고 있다. 특히, MMDBMS는 데이터베이스의 일부 또는 전부를 메모리상에서 관리하도록 하여 디스크 접근을 없애 고성능 트랜잭션이 가능하도록 한 것이며 메인메모리 가격의 하락으로 급격한 발전을 이루고 있다.

<표 1>은 DBMS인 오라클과 MMDBMS인 텔고베이스의 성능 차이를 비교한 것으로 동일한 테이블에 10,000건의 데이터를 입력, 삭제, 수정, 조회하는 데 걸리는 시간(초)으로 비교한 것이다. MMDBMS는 DDBMS보다 33배~108배 정도 빠르게 데이터를 처리하며 데이터가 많아질수록 그 격차는 매우 커졌다[15]. MMORPG에서 데이터베이스에 접속하는 작업이 가장 큰 부하를 차지하므로 MigAgent에 MMDBMS를 도입함으로 부하를 줄일 수 있을 것으로 예상된다.

〈표 1〉 MMDBMS와 DDBMS의 성능 비교 (10,000건의 데이터 처리)[15]

구분	입력(초)	조회(초)	수정(초)	삭제(초)
오라클	20.389	6.269	18.136	19.378
텔코베이스	0.37	0.19	0.341	0.18

4. 구현 및 고찰

4.1 메시지 교환 모델

각 필드서버는 PC가 이동 또는 특정행위를 할 때마다 발생하는 이벤트를 처리해야 한다. 따라서 각 PC의 행동 모델에 따라 게임 시스템 내의 각 서버와 송수신하는 메시지가 달라진다. 본 논문에서는 실험을 위해 각 PC에 서로 다른 이동 속도와 이동 방향 변경 주기 등을 부여하였다. 이렇게 모델링된 PC가 동작하게 되면 MigAgent, 필드 서버 변경, AOI와 AC의 크기 등에 따라 메시지 교환 과정이 달라진다.

<표 2>는 각 PC가 위치한 공간 정보를 바탕으로 이동 여부, AC 출입 여부, 필드 변경 여부들에 따라 구분한 것이고 <표 3>은 이동 여부와는 상관없이 인접 PC들에 대한 동기화 정보의 갱신 여부에 따라 구분한 것이다. <표 2>에서 PC_CHANGE_FIELD의 경우 MigAgent 사용 여부에 따라 동기화 정보 갱신 및 사용자 인증 대상이 데이터베이스와 클라이언트에서 MigAgent로 달라지며 클라이언트와의 연결 재설정 과정이 필요하다. PC_PENDING_TO

〈표 2〉 이동 관련 메시지 교환 모델

Event	설 명
PC_STAY_FIELD	- PC가 동일 필드 내에 머무는 경우 - 데이터베이스에 대한 주기적인 동기화 정보 갱신
PC_STAY_AC	- PC가 동일 필드의 AC 내에 머무는 경우 - 데이터베이스에 대한 주기적인 동기화 정보 갱신 - 필드서버와 MigAgent와의 동기화 정보 갱신
PC_CHANGE_FIELD	- PC의 필드 이동 - 데이터베이스에 대한 동기화 정보 갱신 - 사용자 인증, 연결 재설정
PC_MOVE_TO_FIELD	- PC의 AC 영역 탈출 - MigAgent내 레코드 비활성화
PC_MOVE_TO_AC	- PC의 AC 영역 진입 - MigAgent내 레코드 활성화
PC_PENDING_TO_FIELD	- PC의 AC 영역 탈출 보류
PC_PENDING_TO_AC	- PC의 AC 영역 진입 보류

_FIELD와 PC_PENDING_TO_AC는 PC의 이동 방향과 체류 기간 등의 정보를 활용하여 AC 영역 출입을 최소화시키기 위한 것이다.

〈표 3〉 동기화 관련 메시지 교환 모델

Event	설 명
AOI_NO_AC	필드 영역에 위치한 PC의 AOI가 AC 영역을 포함하지 않음
AOI_NO_FIELD	AC 영역에 위치한 PC의 AOI가 필드 영역을 포함하지 않음
AOI_INCLUDE_FIELD	AC 영역에 위치한 PC의 AOI가 필드 영역을 포함
AOI_INCLUDE_AC	필드 영역에 위치한 PC의 AOI가 AC 영역을 포함
AOI_INCLUDE_NF	PC의 AOI가 인접 필드의 영역을 포함

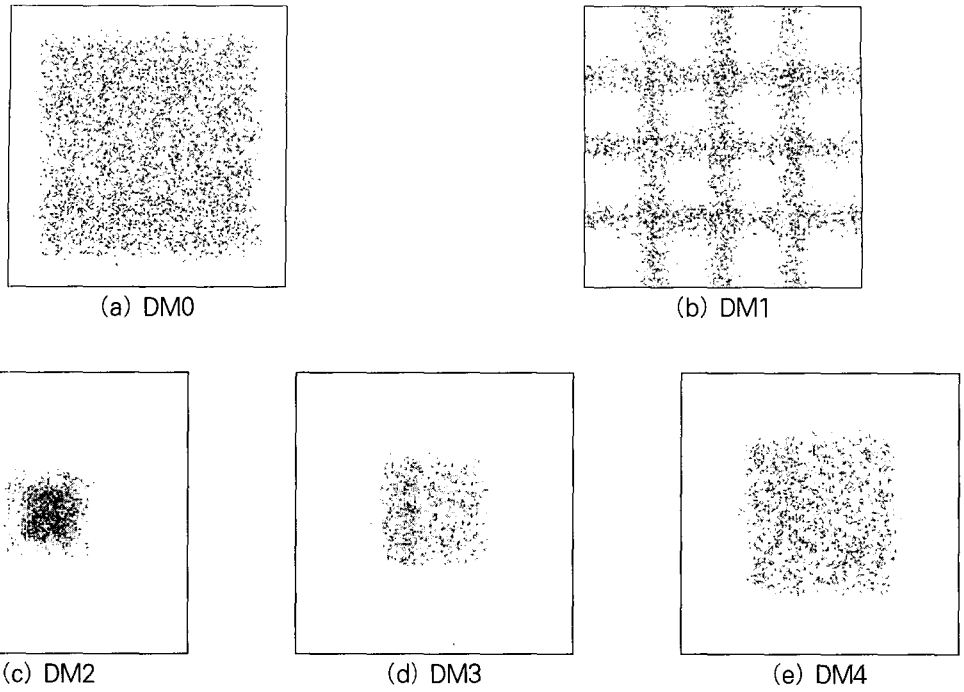
4.2 PC의 초기 배치 모델

시뮬레이션을 위해 PC들의 위치를 초기화하는 방법에 따라 실험 결과는 상당히 많은 편차를 보이므로 PC들의 초기 배치 방법을 미리 구분하여 배치한 후 실험에 활용한다.

<그림 4>에서 DM0은 전체 공간에 고루 배치한 것으로 실제 발생하지는 않겠지만 Seamless 환경에서의 MigAgent 성능을 실험하기 위해 사용한다. DM1은 각 필드의 AC에 PC들을 배치시킨 것이다. DM2, DM3, DM4는 일정 영역(1/16, 1/9, 1/4) 내에 일정 비율 이상의 PC가 몰리는 경우로 특정한 이벤트가 있는 경우에 발생할 수 있는 경우를 추상화한 것이다.

4.3 실 험

본 논문에서는 실험을 위해 2,560 x 2,560의 셀로 구성된 전체 맵을 16개의 필드 서버가 분할 관리하도록 하고 20,000 개의 PC가 이동 속도 및 이동 방향을 랜덤하게 변경하며 진행하도록 하였다. 실험은 턴 방식으로 진행하였다. 매 턴마다 각 PC들의 이동 및 동기화 상태에 따라 발생하는 이벤트(<표 2>, <표 3>)를 측정하여 이벤트들의 가중치를 합산함으로써



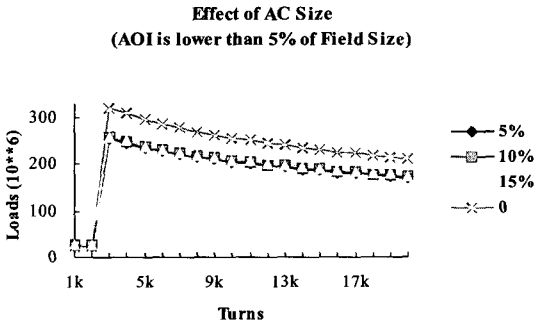
〈그림 4〉 PC의 초기 배치 모델

부하를 측정하였다. 보다 안정적인 결과를 위해 각 설정에 대해 10회씩 실험한 결과의 평균치를 사용하였다.

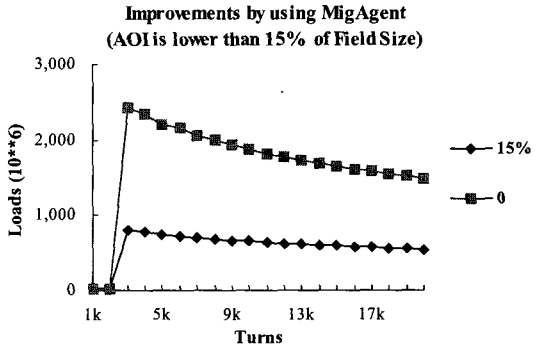
MigAgent을 이용함으로써 얻을 수 있는 성능 향상을 확인하기 위해 각 필드의 AC 크기를 각 필드가 담당하는 영역의 5%, 10%, 15%의 크기로 설정하여 부하를 측정하였다. AC의 크기가 커진다는 것은 MigAgent가 개입하는 영역의 확대를 의미하며 필드서버가 MigAgent에게 수시로 갱신해주어야 할 PC 개체수의 증대를 의미한다.

AOI는 AC의 크기를 초과하지 않도록 하였다. AOI가 AC보다 클 경우 MigAgent를 도입하더라도 인접필드에 포함된 AOI 영역의 동기화 정보를 얻기 위해 MigAgent가 아닌 인접필드서버를 경유해야 하기 때문에 MigAgent의 도입 효과를 상쇄시킬 것이다. AOI의 크기가 커진다는 것은 클라이언트의 랜더링을 위해 동기화 되어야 할 영역의 확대를 의미하며 AOI가 인접필드를 포함하고 있을 경우 상당한 비용 상승을 초래하게 되므로 MigAgent를 사용할 경우 비용 절감을 가져올 것으로 예상된다.

<그림 5>는 AOI를 필드 크기의 5% 이내로 제한시킨 상태에서 AC의 크기를 조절해가면서 부하를 측정한 것으로 AC의 크기가 커질수록 부하 개선 효과가 뚜렷하게 나타남을 확인할 수 있다. AC의 크기가 필드 크기의 5%인 경우 약 17.9%, 10%인 경우 16.2%, 15%인 경우 14.8%의 개선 효과가 있다. 급격하게 부하가 증가하는 부분은 초기에 배치된 PC들의 필드 이동이 대규모로 일어나는 상황으로 분석되며 대규모 이동이 있은 후에는 점차 안정된 부하를 보여주고 있다. AOI의 크기를 늘리면 MigAgent의 도입 효과는 더욱 극명하게 나타난다. AOI의 크기를 10% 이내, 15% 이내로 제한폭을 늘리면 약 28.9%, 57.4%의 개선 효



〈그림 5〉 AC 크기에 따른 부하 변화 (AOI는 필드 크기의 5% 이하)



〈그림 6〉 MigAgent에 의한 개선 (AOI는 필드 크기의 15% 이하)

과를 보인다<그림 6>.

PC의 분포 형태에 따라 부하의 변화를 분석하기 위해 앞에서 기술한 5가지 배치 모델에 따른 부하 변화를 분석하였다<표 4>. <그림 7>은 배치 모델 DM0, DM1, DM4 등에 대한 실험 결과이며 AOI가 커지면 AC내의 동기화 대상 PC의 수가 늘어나게 되므로 개선 효과가 커짐을 확인할 수 있다. 배치 모델 중 DM1의 경우 AC 내에 집중적으로 PC를 배치시켰기 때문에 개선 효과가 가장 컸으며 비교적 큰 영역에 분산되어 있는 DM4에서 그 효과가 가장 작았다.

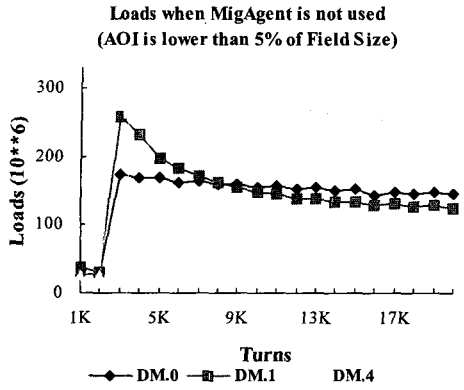
5. 결 론

MMORPG들은 직면한 문제들을 해결하기 위해 상당한 수준의 자원을 필요로 한다. 예를 들어, 가상공간을 유지해야 하고 높은 수준의 하드웨어가 필요하며 시스템 운영 전담 스태프들이 필요하다. 이러한 이슈들을 해결하기 위한 개발자들의 노력에도 불구하고 이 장르의 게임들은 여전히 인구과밀 또는 인구부족, 랙(lag), 지원 부족 등을 안고 있다.

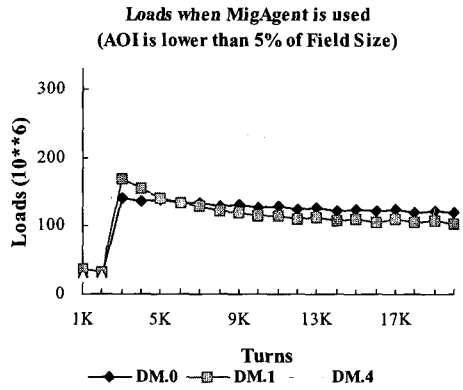
본 논문에서는 Seamless 게임 환경에서의 동기화 메시지 트래픽의 최소화와 이로 인한 체감 성능 저하를 최소화하기 위한 방안으로 MigAgent를 도입한 기법을 제안하고 그 효과를 비교 분석하였다. 각 게임 서버가 관리하는 영역을 Field라 칭하고 이 서버를 필드서버라 한다. 각 필드의 경계면에 위치한 AC 내에 위치한 PC들의 정보를 MigAgent가 관리하도록 한다. 고속의 MMDBMS를 사용하는 MigAgent는 AC 내 PC들의 동기화 정보를 관리하며 인접 필드로의 이동시 PC의 이동 정보를 제공하

〈표 4〉 배치 모델에 따른 부하 개선 효과

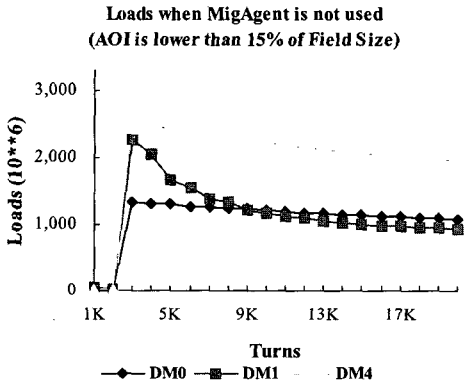
AOI \ 배치 모델	DM0	DM1	DM2	DM3	DM4
필드 크기의 5% 이내	16.3%	19.1%	20.2%	16.4%	14.5%
필드 크기의 10% 이내	28.5%	33.3%	32.1%	27.6%	25.6%
필드 크기의 15% 이내	36.9%	41.3%	39.8%	35.1%	33.1%



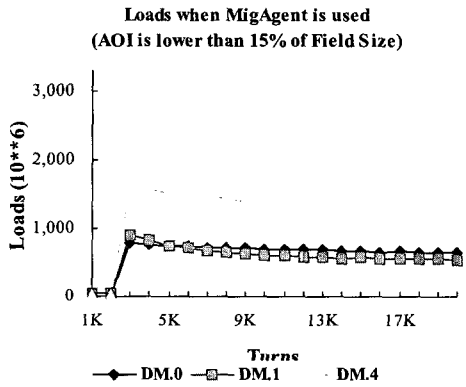
(a) MigAgent를 사용하지 않은 경우
(AOI가 필드 크기의 5% 이하)



(b) MigAgent를 사용한 경우
(AOI가 필드 크기의 5% 이하)



(c) MigAgent를 사용하지 않은 경우
(AOI가 필드 크기의 15% 이하)



(d) MigAgent를 사용한 경우
(AOI가 필드 크기의 15% 이하)

〈그림 7〉 PC 분포 형태별 부하 변화 (AOI는 필드 크기의 15% 이하)

고 각 PC의 AOI가 인접 필드를 포함할 경우 인접 필드의 PC 동기화 정보를 제공함으로써 필드서버의 지연을 최소화시킨다. AC 경계면과 이동 방향과의 방향성 정보와 한계 시간을 이용하여 AC 출입을 최소화함으로써 MigAgent에 의한 불필요한 PC 동기화 정보 로딩 작업을 최소화 하였다. 이 방식은 기존의 구조에 큰 변화를 가져오지 않고 상당한 성능 개선 효과를 보이는 장점이 있다.

실험 결과, AOI를 필드 크기의 5% 이내로 제한시킨 상태에서 AC의 크기를 조절해가면서 부하를 측정한 경우 AC의 크기에 따라 17.9%~15%의 개선 효과가 있었으며 AC와 AOI의 크기가 커질수록 MigAgent에 의한 부하 개선 효과가 뚜렷이 나타났다. 그러나 MigAgent가 관리하는 영역인 AC의 크기가 커질수록 MigAgent의 부담은 커지며 실험에서 나타나지 않는 부수적인 오버헤드가 많이 발생할 것이다. 이를 해결하기 위해 다수의 MigAgent를 두는 방안도 있으나 MigAgent와 필드서버와의 복잡도를 증대시키는 부작용이 있을 수 있다. 따라서 지형적인 장애물을 통한 PC가 활동

할 수 있는 AC 영역을 제한함으로써 한 MigAgent가 관리하는 AC내 PC들의 AOI가 다른 MigAgent의 AC와 겹치지 않도록 하는 절충이 필요할 것으로 예상된다. PC의 이동만 관리하도록 할 경우 필요하지 않으나 AC내에 위치한 PC들의 동기화 정보를 제공하기 때문에 이와 같은 절충이 필요하다.

참 고 문 헌

[1] World of Warcraft Community Site, http://www.worldofwarcraft.com

[2] 이만재, "온라인 게임 엔진 기술 동향", 정보과학회지, 제20권 제1호, pp. 12-18, Jan. 2002.

[3] Massively multiplayer online game, http://en.wikipedia.org/wiki/MMOG

[4] 문성원, "분산 seamless 게임 서버에서의 효율적인 게임 공간 관리 기술", 서강대 정보통신대학원 학위 논문, 02. 2005.

[5] 남재욱, "온라인 게임서버 프로그래밍", 한빛미디어, 05. 2004.

[6] Katherine L. Morse, "Interest Management in Large-Scale Distributed Simulations," UC Irvine, Information and Computer Science Technical Report, ICS-TR-96-27.

[7] Gary Tan et al., "Grid-Based Data Management in Distributed Simulation," Proceedings of the 33rd Annual Simulation Symposium 2000, 2000

[8] Li Zou, Ammar, M.H., Diot, C., "An Evaluation of Grouping Techniques for State Dissemination in Networked Multi-User Games," Proceedings. Ninth International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'01), pp. 33-40, 2001

[9] J. Huang, Y. Du, C. Wang, "Design of the Server Cluster to Support Avatar Migration," Proceedings. Virtual Reality 2003, pp. 7-14, 2003

[10] 박일규, 심광현, 김종성, "온라인 게임서버에서의 k-최근접 관심영역 관리기법", 한국정보과학회 2001년 추계학술대회, vol. 28, no.02, pp.547-549, 2001.

[11] 김용오, "예측로딩을 통한 온라인게임 부하 감소 기법", 한국정보처리학회 2005년 추계학술대회, vol.12, no.01, pp.237-240, 2005

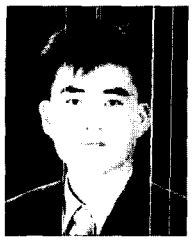
[12] 이철민, 박홍성, "다중 사용자 게임 성능 향상을 위한 데이타 가상 그룹핑 방법", 한국정보처리학회 논문지 B, vol.30, no. 03, pp.231-238, 2003

[13] P. Morillo, J. M. Orduña, M. Fernández, "A comparison study of evolutive algorithms for solving the partitioning problem in distributed virtual environment systems," Parallel Computing, vol. 30, no. 5-6, Amsterdam : Elsevier B. V., pp. 585-610, 2004

[14] 이현진, "프로세스 기반 분산 게임 서버 아키텍처의 연구", 인제대학교 석사학위논문, 02. 2004.

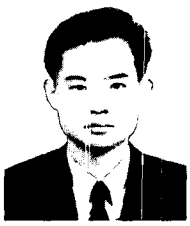
[15] 김상하 "DDBMS와 MMDBMS의 비교 분석", 월간 디지털콘텐츠, 2004.7

◐ 저 자 소 개 ◑



김 법 균 (Beob Kyun Kim)
1994년 전북대학교 컴퓨터공학과 (학사)
1997년 전북대학교 컴퓨터공학과 (석사)
2005년 전북대학교 컴퓨터공학과 (공학박사)
2005년 9월 - 2006년 6월 전북대학교 공학연구원 연구원
전북대학교 BK21-전북지역 전자정보 고급인력 양성사업단 박사후과정
E-mail : bkyun.kim@gmail.com

장 행 진 (Haeng Jin Jang)
1987년 서울산업대학교 전자계산학과 졸업(학사)
1994년 호서대학교 전자계산학과 졸업(석사)
2006년 전북대학교 컴퓨터공학과 (공학박사)
한국과학기술정보연구원 팀장
관심분야: 네트워크, 분산 및 병렬처리, 그리드
E-mail : hjjang@kisti.re.kr



유 강 수 (Kang Soo You)
1991년 전북대학교 컴퓨터공학과 (공학사)
1994년 전북대학교 컴퓨터공학과 (공학석사)
2005년 전북대학교 영상공학과 (공학박사)
1996년~2005년 전주대학교 교양학부 객원교수
2006년~현재 전주대학교 교양학부 교수
관심분야: 영상처리, 분산 및 병렬처리, 컴퓨터그래픽스
E-mail: gsyou@jj.ac.kr