Journal of Korea Game Society 2013 Apr; 13(2): 39-48
http://dx.doi.org/10.7583/JKGS.2013.13.2.39

게임 프로그래밍

연산서버를 적용한 MMORPG 게임서버에 관한 연구

배성길*, 김혜영**
홍익대학교 일반대학원 게임학과*, 홍익대학교 게임학부 게임소프트웨어 전공**
ddinggill@naver.com, hykim@hongik.ac.kr

A Study on the MMORPG Server Architecture Applying with Arithmetic Server

Sung-Gill Bae*, Hye-Young Kim**
Major in Game Software, Graduate School, Hongik University*
Major in Game Software, School of Games, Hongik University**

요 약

가상공간에서 대규모 게임 사용자들 간의 상호작용이 활발히 일어나는 MMORPG(Massively Multi-player Online Role-Playing Games)에서는 다수의 클라이언트의 접속 요청 및 작업 요청을 실시간으로 빠르게 처리할 수 있어야 한다. 그러나 클라이언트의 접속자 수가 늘어날수록 처리해야 할 작업량이 많아지며, 게임서버의 부하도 높아지게 된다. 이를 해결하기위해 많은 개발자들은 분산서버구조를 적용하여, 동적 맵 분할, 서버의 기능에 따른 부하 분산 등의 기법들을 제시하고 있다. 현재 대부분의 MMORPG 게임서버는 하나의 월드를 Zone방식으로 나누어서 각각의 영역을 다수의 게임서버가 담당하여 게임을 진행하고 있다. 이러한 방식은 사용자의 빈번한 서버이동에 따른 데이터 갱신 등의 오버헤드를 발생하여 게임서버에 큰 부하를 주고 있다. 따라서 본 논문에서는 게임서버의 부하를 줄이기 위해 데이터의 연산을 담당하는 연산서버를 적용하여 기존 게임서버의 효율은 높이고 더 많은 사용자의 접속과 작업을 처리할 수 있는 구조를 제안하고, 수학적인 모델링과 성능분석을 통해 기존 연구들과의 비교 시의 제안 기법의 효율성을 보였다.

ABSTRACT

In MMORPGs(Massively Multi-player Online Role-Playing Games) a large number of players actively interact with one another in a virtual world. Therefore MMORGs must be able to quickly process real-time access requests and process requests from numerous gaming users. A key challenge is that the workload of the game server increases as the number of gaming users increases. To address this workload problem, many developers apply with distributed server architectures which use dynamic map partitioning and load balancing according to the server function. Therefore most MMORPG servers partition a virtual world into zones and each zone runs on multiple game servers. These methods cause of players frequently move between game servers, which imposes high overhead for data updates.

In this paper, we propose a new architecture that apply with an arithmetic server dedicated to data operation. This architecture enables the existing game servers to process more access and job requests by reducing the load. Through mathematical modeling and experimental results, we show that our scheme yields higher efficiency than the existing ones.

Keyword : MMORPG, Distributed Server, Game User

Received: Nov. 19, 2012 Revised(1st): Jan. 03, 2013
Revised(2nd): Feb. 22, 2013 Accepted: Feb. 28, 2013
Corresponding Author: Hye-Young Kim(Hongik Hongik University)
E-mail: hykim@hongik.ac.kr

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

MMORPG(Massively Multiplayer Online Role-Playing Game)의 성장세가 뚜렷하고 시장이 확대됨에 따라 게임 사용자의 수가 증가함으로써 하나의 서버에서 처리 할 수 있는 사용자의 수와 데이터의 양의 한계는 중요한 문제가 되고 있다. 이를 해결하기 위해 분산 서버 방식의 구조가 사용되고 있는데, 온라인 게임서버는 사용자들의 그룹형성이나 사용자들이 게임을 즐길 수 있는 환경을 제공하기 위한 게임의 맵, NPCAI 등의 데이터 처리하는 역할까지 수행해야 한다[1]. 이 게임서버의 역할은 처리해야하는 게임데이터, 사용자 관리 등의 수행해야하는 게임서버의 기능 등의 기준으로 나누어져 효율적으로 수행할 수 있게 설계되고, 확장성도 고려해야 한다.

MMORPG를 서비스하기 위해서는 하나의 커다란 맵을 여러 개의 게임서버가 담당하는 영역으로 나누어서 게임플레이 및 사용자의 처리를 담당하게 되는데 이는 결국 사용자의 수와 네트워크 트래픽의 제한으로 인해 게임서버가 담당하는 영역이 제한됨에 따라 게임서버의 수가 늘어나게 되고 이를 통해서 게임 내에서 사용자가 맵을 이동할 시 게임서버 간 사용자의 이동이 빈번하게 발생하게 되는데 이러한 점은 게임서버에 데이터를 처리하는 것보다 큰 부하를 주고 있다[2,11].

본 논문에서는 사용자의 이동이 빈번하게 발생하는 것을 최소화하기 위해 기존 게임서버의 기능 중 하나인 데이터 연산을 담당하는 연산서버를 적용하여 게임서버의 부하를 줄이고 게임서버가 담당하는 영역과 사용자의 수를 늘림으로써 효율적인 게임서버를 제안한다.

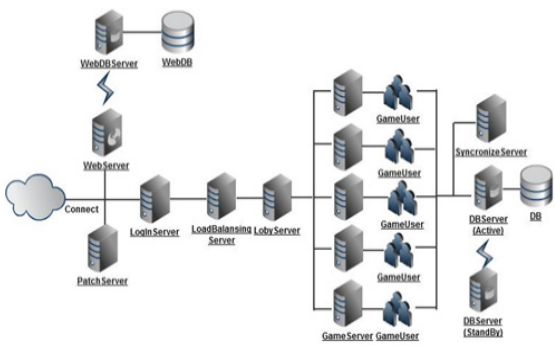
2. 관련연구

우수한 기술의 지원으로 인해 네트워크를 이용한 온라인게임들은 다양한 형태로 거대한 배경을 기반으로 발전하고 있고 게임을 동시에 플레이 하는 사용자 수 또한 증가함에 따라서 게임서버 1대가 모든 일을 하는 단일 서버구조로는 현재의 대규모 사용자 처리하기에는 불가능해졌다. 이를 해결하기 위해서 [Fig. 1]과 같은 게임서버가 수행하는 기능들을 각각의 서버로 독립적으로 분산 운용하여 각 기능을 서버가 담당하게 하는 분산서버 구조가 대세를 이루고 있다. 기능적으로 로그인/업데이트서버, 통신서버, 게임서버, 데이터베이스 서버 등으로 분산 시킨 후 서버들 간의 통신을 통해 게임을 처리하는 구조다[3].

기존의 온라인 게임서버의 구조는 [Fig. 1]과 같다. 사용자들은 최초에 로그인서버에 접속하게 되고 사용자의 최종적인 위치를 데이터베이스 서버를 통해서 가져온 뒤 각 위치에 알맞은 게임서버에 사용자들을 할당해준다. 그리하면 사용자들은 자신의 위치에 따른 게임서버에 접속을 하게 되고 이를 통해서 게임을 진행하게 된다.

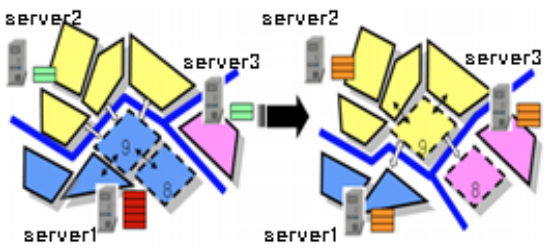
게임서버는 접속한 사용자의 직접적인 게임플레이 환경을 제공하고 네트워크를 통해 게임을 진행하고 자신이 담당하는 맵의 영역과 사용자들을 관리하여 데이터를 처리하게 된다[3,4].

게임을 진행하면서 사용자는 자신의 서버가 담당하는 영역을 벗어난 위치로 이동을 하게 되는데 이때 게임서버는 정보를 사용자가 새롭게 이동한 영역을 담당하는 게임서버에서 넘겨주고 새로운 게임서버가 사용자의 게임플레이를 담당하게 되는데 이러한 사용자 이동 작업은 서버에 큰 부하를 준다.



[Fig. 1] Present distributed server model

이러한 구조에서는 게임서버에 몰리는 사용자의 수와 네트워크 트래픽의 제한으로 게임서버가 담당하는 영역이 점점 줄어드는 반면, 게임서버의 수가 늘어나게 되고 담당하는 영역 또한 작아진다. 그러면 사용자가 게임을 플레이 하는데 있어서 작은 위치의 이동만 있더라도 빈번한 사용자의 서버 이동이 발생하게 되고 이는 서버를 효율적으로 사용하지 못하는 상황이 발생하게 된다. 이를 해결하기 위해 [Fig. 2]의 방법으로 맵 부하 분산 서버를 이용하여 게임서버가 담당하는 영역을 동적으로 변환해주며 불균등한 게임 배경 점유에 따른 각 서버 부하의 불균형을 제어하는 방법으로 각 서버의 효율을 높이는 방법이 있으나 서버 간 맵 동적변환이 이루어지게 되는 기준이 게임마다 너무나 다르고 부하도 크다[5,6].



[Fig 2] Dynamic allocation by map balancing

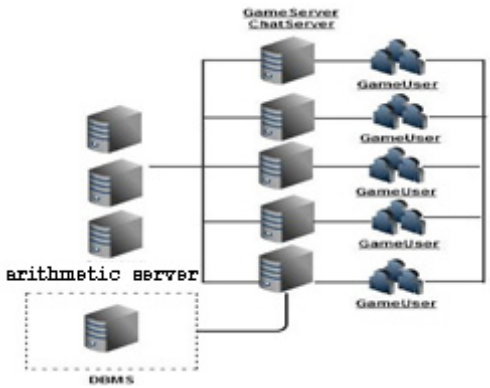
다른 방법으로는 통신서버라는 사용자 접속과 게임처리와 실질적으로 관계가 적은 채팅부분과 같은 메시지를 처리(게임서버가 직접 관여할 필요 없는 인스턴스 메시지)하는 서버가 있는데 이는 사용자의 맵 이동 시 접속처리 부분만을 처리하기 때문에 게임에 직접적인 영향을 끼치게 되는 즉, 게임서버에 직접적으로 부하를 주는 네트워크 트래픽을 감당하는 데는 제한이 있다. 그리고 클라이언트-서버 간 통신이 통신서버를 무조건 거쳐서 가기 때문에 모든 패킷의 전달에 있어서도 2단계 과정을 거치기 때문에 비효율적인 방법이다.

이를 해결하기 위해서 본 논문에서는 통신에는 직접적으로 관여하지는 않지만 게임서버의 연산부분을 처리하는 연산서버의 개념을 두어 게임서버의 직접적인 부하를 줄이는 방법을 제안한다.

3. 제안기법

본 논문에서 게임서버의 작업부하를 효율적으로 나누어 게임서버의 연산 작업 중 일부분을 연산서버 에서 처리함으로써 보다 효율적인 게임서버 구조를 제안한다.

3.1 연산서버의 작업을 위한 메시지 분배작업



[Fig. 3] Server architecture applied with Arithmetic server

[Fig. 3]에서와 같이 게임서버와 내부적으로 연동되는 연산서버를 두어 작업을 분해하는 방법이다. 이를 위해 기존 게임 유저에 의해 요청되는 작업 메시지의 유형을 게임서버가 직접적으로 처리해야하는 메시지와 연산서버를 이용해 작업을 처리할 수 있는 메시지로 구분하는 작업이 필요하다. 로그인 서버는 유저의 접속을 처리 한 뒤 유저가 원하는 게임서버로 연결시켜 준 뒤 그 유저는 자신의 게임서버로 연결되어 자신의 정보도 게임서버에서 관리한다. 게임서버의 전체 월드는 Zone방식으로 구분되고 유저가 자신의 게임서버가 담당하는 Zone을 벗어나게 되면 다른 게임서버로 이동이 되는 것이다[7]. 이러한 유저의 접속과 이동을 처리하는 작업 요청 메시지들은 연산서버를 이용해서

하는 것보다 직접적으로 게임서버에서 처리하는 것이 게임서버 자체가 사용자 정보를 가지고 있기 때문에 더 효율적이므로 이러한 작업을 연산서버에서 담당할 필요는 없다.

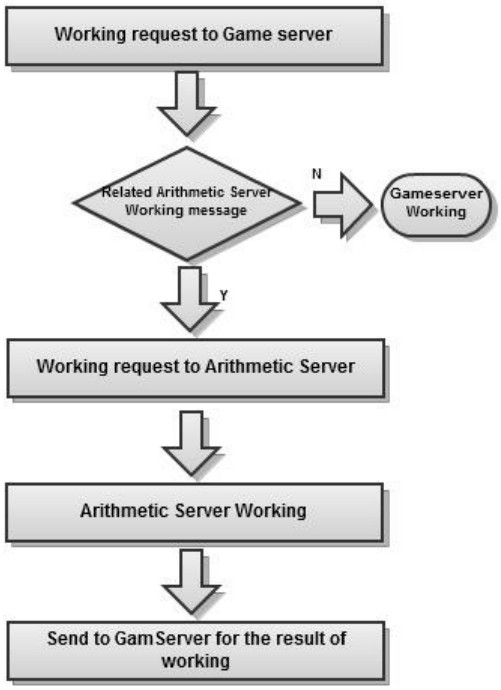
게임을 진행하는데 있어 필요한 다양한 작업 메시지들의 연산을 연산서버에서 담당할 수가 있는데 일반적인 게임에서의 작업요청 메시지 중 연산서버가 담당 가능한 메시지와 불가능한 메시지를 구별을 [Table 1]에서와 같이 구별할 수 있다[8,9,12].

[Table 1] Classified Working Message

Classification	Working Message
Possible responsible message	1. movement of Character 2. message related to AI 3. message related to battle 4. message related to Item 5. message of character's position revision 6. movement of Monster
Impossible responsible message	1. Access of Game Server by User 2. Movement of User between Game Servers 3. Chat message 4. message related to grouping

3.2 메시지 부하 분석을 통한 연산서버의 활용

위에서 분류한 메시지 종류를 바탕으로 연산서버가 담당할 수 있는 작업 중 게임서버에서 직접적으로 처리하는 서버구조의 속도와 게임서버와 연산서버와의 통신시간과 작업처리하는 시간을 비교하여 연산서버를 도입하여 처리하는 작업이 더 효율적인 메시지 작업들을 연산서버가 담당하게 되는 방법으로 서버 구조가 구성된다. 유저의 요청 작업 중 실시간으로 빠른 응답이 요구되는 작업들은 게임서버에서 처리하는 것이 효율적이고 연산 등에 소요되는 시간은 길지만 빠른 응답이 요구되지 않아도 되는 작업들을 연산서버에서 담당하도록 하였다.



[Fig. 4] Process of proposed scheme

[Fig. 4]에서 보는 것과 같은 작업 과정을 거쳐 유저의 작업 요청 작업을 수행한다.

3.3 효율적인 작업 메시지 선택

게임 진행시 발생되는 수식들의 계산을 위해서 연산서버가 담당할 수 있는 메시지 중에서 기존 서버구조와 비교 했을 시 가장 효율적인 메시지를 구별하기 위해 각 메시지마다 기존 서버구조에서 걸리는 총 수행시간과 본 논문에서 제안하는 기법에서 걸리는 총 수행시간을 비교하여 연산서버가 작업을 담당할 메시지를 선택한다.

효율적인 작업 메시지의 조건은 [Table 2]와 같다.

[Table 2] Efficient message condition

1. many request message 2. To minimize database access 3. Spent a lot of time working operation 4. minimizes the data transfer of the operations server task required upon request

3.1절에서 분배한 작업 중 사용자의 이벤트에 대한 응답이 바로 전달되어져야 하는 전투 관련 메시지와 게임서버와 연결된 데이터베이스 서버를 거쳐야 하는 아이템 관련 메시지는 연산서버가 작업을 담당하는데 있어서 효율적이지 못하다. 그러나 사용자가 비전투시 이동하는 이동 동기화 관련된 메시지 같은 경우 이벤트 요청은 계속해서 요청이 많지만 요청 시 마다 전투이벤트 같은 즉각적인 반응이 아닌 어느 정도의 시간마다 서버에서 처리해주면 되는 메시지이다.

[Table 2]에서의 조건을 가장 만족하는 사용자의 캐릭터 이동 동기화 관련 메시지와 위치보정 메시지, 몬스터 이동 동기화 관련 메시지들을 연산서버에서 선택하여 게임서버 대신에 작업을 수행한다. 사용자가 작업을 요청하는 패킷을 게임서버로 보내면 게임서버에서는 패킷 분류를 하여 연산서버로 작업할 패킷들은 연산서버로 변환된 작업 요청 패킷을 보낸다.

4. 성능 분석

본 논문에서는 작업 요청 패킷을 발생시키는 가상의 게임 사용자들의 접속을 지속적으로 발생시켜서 기존연구와 제안 기법과의 서버 효율을 비교하였다. 성능평가를 위해 지속적으로 접속하는 사용자 수를 증가하면서 기존의 서버 모델과 연산서버를 적용한 서버 모델의 성능을 평가하고, 게임서버는 두 모델 다 같은 성능의 게임서버를 하나씩 사용하였고 제안하는 모델에서는 연산서버를 추가하여 진행하였다.

서버의 모델은 IOCP방식의 모델을 사용하였으며 데이터베이스는 MSSQL을 서버와 ODBC로 연동하여 구현하고 서버에서는 실시간으로 로그를 분석하였다[10].

[Table 3] Server execution time

Classification	Gameserver	Arithmeticserver + Gamerver
Working time	T_{wg}	T_{wc}
Communication time	T_n	$T_n + T_{cg}$
Total execution time	$T_{wg} + T_n$	$T_{wc} + T_n + T_{cg}$

4.1 사용자 수에 따른 게임서버 작업시간 분석

게임서버와 클라이언트 간의 분석패킷의 설계는 패킷의 전체 사이즈와 타입을 알려주는 패킷 헤더 부분과 실 데이터 부분으로 구성되었다. 캐릭터의 이동 동기화 관련 메시지 패킷을 게임서버에 요청하는 클라이언트 생성하여 게임서버에 요청하는 방식으로 진행하였다. 게임서버가 처리하는 전체 메시지에서 일반 메시지의 비중은 20%정도이다[8]. 기존의 게임서버만을 이용한 모델에 비해 이동 동기화 관련 메시지를 연산서버에서 처리하는 제안모델의 작업시간은 기존 모델의 시간보다 80%정도의 시간만이 걸렸다. 이는 기존모델은 게임서버에서 사용자 접속처리 등의 다른 작업에도 부하가 걸리기 때문이다.

C_m은 접속자의 수, P_w는 서버가 처리하는 전체 메시지의 수라 하고 t는 각 메시지의 작업을 처리하는데 걸리는 시간을 나타내며 전체 사용자의 수에 따른 작업시간은

$$T_{wg} = C_m * P_w * t \quad \text{.....(eq. 1)}$$

전체 사용자의 수에 따른 연산서버를 적용한 제안기법의 작업시간은

$$T_{wc} = C_m * P_w * 0.8 * t \quad \text{.....(eq. 2)}$$

가 된다.

게임서버만을 이용한 총 수행시간을 T_1이라하고, 연산서버를 이용한 총 수행시간을 T_2, 게임서버와 사용자간의 통신시간을 T_n, 그리고 게임서버와 연선서버의 통신시간을 T_{cg}라고 할 때, 게임서버만을 이용한 총 수행시간은 (eq. 1)을 통해서

$$T_1 = T_{wg} + T_n \quad \text{.....(eq. 3)}$$

연산서버를 이용한 총 수행시간은 (eq. 2)를 통해

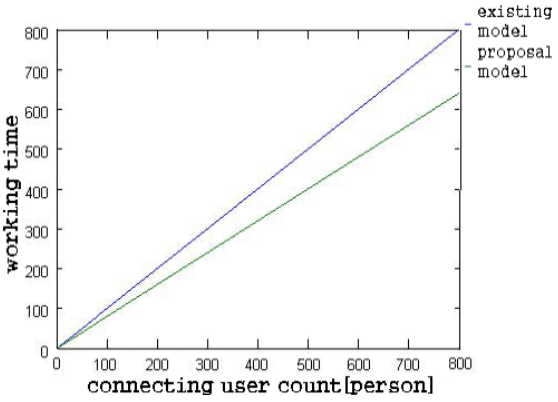
$$T_2 = T_{wc} + T_n + T_{cg} \quad \text{..... (eq. 4)}$$

가 된다.

따라서 기존의 연구와 제안 기법에서의 작업시간, 통신시간, 그리고 총수행시간은 [Table 3]과 같이 나타낼 수 있다.

[Fig. 5]에서와 같이 게임에 접속하는 사용자 수를 증가시키면서 기존연구와 제안기법의 게임서버에서만의 작업시간을 비교하였다. 기존연구의 게임서버는 게임 사용자의 작업요청 처리를 게임서버 자체적으로 처리하고 제안기법의 게임서버는 사용자의 작업요청 처리를 전부 하는 것이 아닌 연산서버와 나누어 작업을 하기 때문에 두 모델의 게임서버 자체만의 작업처리에 걸리는 시간은 접속하는 게임 사용자의 수가 증가 할수록 기존연구와 제안 기법과의 작업시간차가 커지게 된다. 즉 접속하는 게임 사용자의 증가수가 똑같더라도 게임서버의 자체에 걸리는 부하는 기존 연구의 기법에서는 더 많은 부하가 발생하기 때문에 제안기법이 더욱 효율적임을 알 수 있다.

[Fig. 6]에서 보듯이 접속하는 사용자의 수가 증가하면 할수록 연산서버를 도입한 모델과 이전 모델과의 게임서버의 작업시간은 점점 커짐을 알 수 있다.



[Fig. 5] Comparison of execution time

4.2 유저 수에 따른 통신시간 분석

연산서버를 적용한 제안 기법에서의 게임 서버와 연산서버간의 통신에 대한 오버헤드가 미치는 영향을 분석하기 위해 게임서버와 사용자간의 통신시간과 게임서버와 연산서버간의 통신시간을 분석하였다. 대부분 게임서버와 사용자의 통신시간은 20ms 정도면 양호한 네트워크 상태라고 본다[5]. 따라서 게임서버와 사용자간의 통신 속도는 20ms로 일정하게 유지되게 가정하였고 게임서버와 연산서버는 내부적인 통신망으로 이루어져있는 테스트 환경에서 접속하는 게임 사용자의 수를 증가시키면서 게임사용자와 게임서버간의 통신시간 및 게임서버와 연산서버와의 통신시간을 측정하여 [Table 4]에 나타내었다.

[Table 4]와 같이 접속하는 게임사용자의 수가 증가하더라도 게임서버와 연산서버의 통신시간은 크게 늘어나지 않고 일정하게 유지되는 것을 알 수 있다. 이는 게임서버와 연산서버의 실험환경이 LAN기반의 충분한 대역폭이 지원되고 서버의 수가 제한적이었기 때문이다.

[Table 4] Communication time

User (person)	Commutication time(ms) (server<->user)	Communication time(ms) (Game server<-> Arithmetic server)
100	20	1
200	20	1
300	20	1
400	20	1
500	20	1
600	20	1
700	20	1
800	20	2

4.3 서버 총 수행시간 분석

[Table 5]에서 보듯이 보통 서버와 사용자의 통신시간은 20ms 정도면 양호한 상태인데 사용자의 수가 늘어나고 작업 요청 패킷의 양이 많이 질수록 서버의 작업시간이 늘어남에 따라서 서버와 사용자의 통신시간은 일정하더라도 서버의 총 수행시간은 점차적으로 늘어나고 있다.

반면에 [Table 5]에서 보듯이 연산서버의 통신시간은 일정하였지만 서버의 작업시간이 처음 사용자가 300명 구간까지는 작업시간이 기존의 구조가 더욱 효율적이었다.

[Table 5]

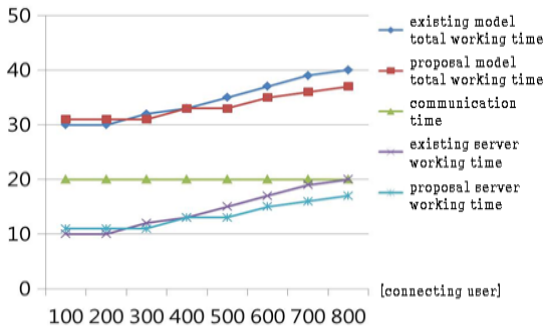
(a) Execution time of previous server

User (person)	Communication time(ms) (srver-user)	Server working time(ms)	Total execution time
100	20	10	30
200	20	10	30
300	20	12	32
400	20	13	33
500	20	15	35
600	20	17	37
700	20	19	39
800	20	20	40

(b) Execution time of proposed scheme

User (person)	Communication time(ms) (srver-user)	Server working time(ms)	Total execution time
100	20	11	31
200	20	11	31
300	20	11	31
400	20	13	33
500	20	13	33
600	20	15	35
700	20	16	36
800	20	17	37

400명 구간부터는 서버의 작업수행시간의 증가량이 기존연구보다 점차적으로 좋아지기 시작하였다. 즉, 게임 사용자 수가 증가할수록 서버의 총 수행시간의 차이는 점점 커짐을 알 수 있다. 성능분서 시에 서버 작업시간에는 게임 서버와 연산서버 간의 통신시간도 포함하였다. 유저의 수가 800명인 상황에서의 서버의 총 수행시간의 차이는 3ms로 큰 수치의 차이는 아니지만 실제 상용화된 MMORPG의 접속 유저 수는 800명과는 비교할 수 없이 몇 천, 혹은 몇만명으로 기하급수적으로 늘어나게 되는데 그렇게 된다면 [Table 5]에서 나타나는 기존모델과 제안기법의 차이는 더 크게 발생됨을 예상할 수 있다.



[Fig. 6] Performance Comparison of server models

[Fig. 6]과 같이 서버 사용자와의 통신시간을 일정하게 유지하더라도 사용자의 수가 증가할수록 본 논문이 제안한 기법이 더욱 효율적임을 알 수 있다.

5. 결론 및 향후 과제

본 논문에서는 기존의 MMORPG에서 사용되던 분산서버구조에서 사용자가 서버에 몰림으로써 발생되는 작업의 처리 양에 따른 작업능률 저하를 줄이기 위해 연산서버를 적용하여 사용자의 증가에 따른 작업에 효율적으로 처리함으로써 효율적인 서버 구조를 제안하였다. 기존모델의 게임서버에서 처리하던 작업을 연산서버에서 나누어 작업하여 게임서버의 작업처리량을 감소시켰다. 즉, 게임서버의 작업시간을 향상시켜 전체적인 서버의 총 수행시간이 효율적인 모델을 제안하였고 이는 사용자의 수가 증가할수록 더 나은 모습을 보였다.

제안 기법에 대한 성능분석에서 구현한 환경이 게임서버와 연산서버가 각 각 하나씩만 존재하고, 패킷의 종류도 MMORPG에 쓰이는 몇 가지의 제한적인 메시지만을 포함하고 있다. 그러므로 더욱 효율적인 검증을 위해서는 상용화된 MMORPG 게임서버의 패킷 분석을 통해 게임서버와 연산서버의 작업 분배를 실험함으로써 신뢰도를 높일 수 있을 것으로 예상한다.

또한, 연산서버와 게임서버와의 위치를 내부의 지역적인 환경이 아닌 외부 인터넷 환경에서의 성능 비교도 필요할 것으로 보인다.

향후 연구로는 또한 제안 기법에서의 연산서버에서의 처리 할 패킷을 자동으로 분별하는 알고리즘을 신뢰성 있도록 제안하고, 연산서버의 작업 수행 능력을 위한 최적화 알고리즘들을 제안하며, 데이터베이스 서버와의 연동을 고려하여 최적화된 게임서버를 제안하고자 한다.

ACKNOWLEDGEMENTS

This research was supported by the MKE(The Ministry of Knowledge Economy), NHN Corp. under IT/SW Creative research program supervised by the NIPA(National IT Industry Promotion Agency) (NIPA-2012-H0506-12-101).

This work was supported by 2013 Hongik University Research Fund.

REFERENCES

[1] Jung-Yeoul Lim and three others “Technical Trend of Distributed Game Server”, Electronics and Telecommunications Trends, Vol. 20, No. 4, pp93-102, 2005

[2] Nam-Kyeong Um, Hyung-Jin Moon, Sang-Ho Lee, “Design and Implementation of A Load Balancer Based on Load Equality between Game Servers”, The Journal of the Korean Institute of Information and Communication Engineering, Vol. 32, No. 4, 2007

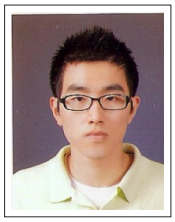
[3] Su-Min Jang, Jae-Soo Yoo, “An Efficient MMORPG Distributed Game Server”, Journal of Korea Contents Association, Vol. 7, No. 1 2007

[4] Kwang-Ho Yang and four others, “Technical Trend of Online Game Server”, Electronics and Telecommunications Trends, Vol. 16, No. 4, pp14-22, 2001

[5] Su-Min Jang, Jae-Soo Yoo, “Efficient Distributed Processing Scheme for Load Balancing of MMORPG Server”, Journal of Korea Contents Association, Vol. 7, No. 11, pp69-75, 2007

[6] Jong-Gwan Choi, Hye-Young Kim, Woo-Sik Woo, “A Study of a Game User Oriented Load Balancing Scheme on MMORP”, Journal of Korea Game Society, Vol. 12, No. 3, pp69-76, 2012

[7] Soon-Gohn Kim, Nam-Jae Lee, Seung-Weon Yang, "A Management method of Load Balancing among Game Servers based on Distributed Server System Using Map Balance Server", Journal of Korea Navigation Institute, Vol. 15, No. 6, pp1034, 2011
[8] Seung-Weon Moon, Hyung-Jae Jo, "A Study on Synchronization Distribution of Server Message in Online Games", Journal of Korea Game Society, Vol. 9, No. 2, 2009
[9] Dong-Hoon Han, "Online Game Server programming", Information Publishing Group, 2007
[10] Hye-Young Kim, Moon-Sung Kim, Dae-Hyun Ham, "A Study of Object Pooling Scheme for Efficient Online Gaming Server", Journal of Korea Game Society, Vol. 9, No. 6, 2009
[11] Santosh Kulkami "Badumna Network Suite A Decentralized Network Engine for Massively Multiplayer Online Applications" Peer-to-Peer Computing, 2009, P2P '09. IEEE Ninth International Conference 2009
[12] Carlos Eduardo Benevides Bezerra and one person, "A load balancing scheme for massively multiplayer online games", Journal Multimedia Tools and Applications archive Vol. 45, Issue 1-3, 2009년



배 성 길 (Bae, Sung Gill)

2006.2.25 홍익대학교 게임소프트웨어학과 이학사
2012.3.2-현재 홍익대학교 일반대학원 게임공학과

관심분야: 게임서버, 분산서버, 게임제작

김 혜 영 (Hye-Young Kim)

2005.2 고려대학교 컴퓨터학과 이학박사
2005.3-2006.8 Wright State Uni. Post-Doc.
2007.3-현재 홍익대학교 게임학부 게임소프트웨어전공 부교수

관심분야 : 모바일 게임, 온라인 게임서버, 게임엔진