Journal of Korea Game Society 2017 Dec; 17(6): 165-172
http://dx.doi.org/10.7583/JKGS.2017.17.6.165

게임프로그래밍

멀티 플랫폼 시뮬레이션 네트워크 게임을 위한 서버 프레임워크 연구

권순정
서강대학교 평생교육원 MTEC
iamsam@sogang.ac.kr

A Study on the Server Framework for Multi-platform Simulation Network Game

Soon-Jung Kwon
Sogang University Continuing Education Center MTEC

요 약

다양한 플랫폼에 따라 게임을 개발하는 데에는 중복되는 작업들이 있는 것 같다. 각 플랫폼에 따라 데이터를 관리하고 처리하는 유사한 작업이 포함되는데, 이는 개발기간에도 영향을 미친다. 본 논문은 최근 유행하고 있는 전투 시뮬레이션 게임을 멀티 플랫폼 기반으로 서버를 설계 하고 제작하여 플랫폼 형태와 상관없이 동일한 서버와 같은 전투 시뮬레이션의 결과를 볼 수 있는 서버 프레임 워크를 제안한다. JSON 데이터 포맷을 사용하여 속성-값 쌍으로 이루어진 프로토콜을 사용하여 디버깅하기 편하게 설계 하였다. 프로그래밍 언어에 독립적이기 때문에 다양한 언어와 통신을 할 수 있는 장점도 가지고 있다. 서버는 아마존 서버를 이용하여 쉽게 서버 구조를 확장하거나 서버 스펙을 업그레이드 할 수 있는 모델을 제안한다.

ABSTRACT

Some duplicate processes are happen in developing games under the diverse platforms. Implementing functions. like processing and managing data, in every platforms have an influence on game development. In this paper, we propose a multi-platform server framework that can simulate combat games such as one server on any platform. It was designed to be easy to debug using a protocol consisting of attribute-value pairs by the JSON data format. Since it is independent of the programming language, it has the advantage of being able to communicate with various languages. The server proposes a model that can easily upgrade the structure or the specification using the Amazon web server.

Keywords : Node.js, AWS, Socket.IO

Received: Nov. 12. 2017 Revised: Dec. 15. 2017
Accepted: Dec. 20. 2017
Corresponding Author: Soon-Jung Kwon(Sogang University Continuing Education Center)
E-mail: iamsam@sogang.ac.kr
ISSN: 1598-4540 / eISSN: 2287-8211

Ⓒ The Korea Game Society. All rights reserved. This is an open-access article distributed under the terms of the Creative Commons Attribution Non-Commercial License (http://creativecommons.otg/licenses/by-nc/3.0), which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

1. 서 론

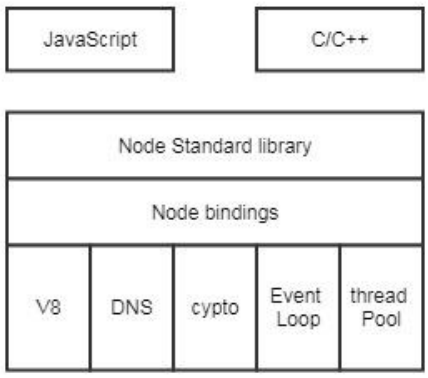
최근 모바일 게임과 웹 게임이 유행하고 있으며 기존의 MMORPG와 같이 역할 수행게임을 하는 것이 아닌 캐릭터들에게 역할을 주고 육성하며 전투를 즐기는 시뮬레이션 게임이 증가하고 있다. <소녀전선>과 같은 게임은 전투 시뮬레이션이 게임성에 많은 영향을 미친다. 시뮬레이션 게임은 단순한 계산으로만 진행이 되지 않고 복잡한 형태의 캐릭터 배치와 중요 변수들을 통하여 게임의 승패를 가지고 결과를 내는 형태의 게임이다. 그리고 기존의 게임을 이용하여 IP만 새로 구매한 후 게임을 다시 제작하는 경우도 있다. 대표적인 게임으로는 <오션 앤 엠파이어>와 <캐리비안의 해적 : 전쟁의 물결>이 있다. 이 두 게임은 같은 전략 전투 시뮬레이션 게임으로 표현 방식이 조금 다를 뿐 같은 구조의 형태를 가지고 있다[1,2].

이와 같은 게임들의 개발 기간 단축과 레벨 디자인 및 게임 밸런싱의 신뢰성을 재활용 할 수 있기 때문에 기존 게임의 데이터들을 그대로 사용한다. 그리고 같은 게임 서버 프레임 워크를 이용하여 여러 가지의 클라이언트를 연결할 수 있다. 이에 따라 지속적인 콘텐츠 추가와 기획 운영을 할 수 있는 DB 구조 및 서버 프레임 워크가 필요하다[3]. 이 논문에서는 이러한 DB 구조를 가질 수 있는 형태의 DB와 전투 시뮬레이션 프레임 워크를 제안한다. 논문의 구성은 다음과 같다. 2장에서는 본 논문에서 사용한 소프트웨어와 모듈에 대해서 설명하고 3장에서는 2장의 도구를 이용하여 구현한 구현 결과를 설명한다. 4장에서는 본 구현 내용에 대한 성능 분석을 하였고 5장은 결론을 기술한다.

2. 연구 배경 및 관련 연구

2.1 Node.js 서버

Node.js 는 구글의 크롬 V8 엔진을 기반으로 한 고성능 네트워크 서버이다. Single Thread 기반의 비동기 IO 처리를 하는데, 자바스크립트로 구성 할 수 있어 생산성 및 접근성이 높다. 비동기 IO 요청으로 CPU가 IO 응답을 기다리는 시간이 필요 없고, 요청이 끝나고 이벤트를 받아 처리하기 때문에 고성능의 퍼포먼스가 구현된다. Node.js는 [Fig. 1] 과 같이 구성되어 있다. 많은 Connection을 동시에 처리하는 시나리오 서버에서도 기존 APM 형태의 서버보다 성능이 우수하다.



[Fig. 1] Node.js configuration diagram

2.2 Socket.io 소켓 통신

웹의 변화로 인하여 클라이언트에서만 요청하는 경우가 아닌 양방향성의 웹 사이트가 점점 늘어나고 있어 이에 대응하기 위해 Socket.IO 가 만들어졌다. Socket.IO 는 자바스크립트 모듈로, HTTP 클라이언트로 푸쉬 메시지를 보낼 수 있다. 다양한 브라우저를 지원하고 사용 편의성이 높다. 기존에는 Polling 방식으로 접속 유지 체킹과 실시간으로 반영되는 게임 화면들을 보여준다. 서버는 정해진 Polling 주기로 클라이언트에서 요청을 받고 해당 요청을 처리한다. Polling 의 단점은 클라이언트가 많아질수록 서버의 부하가 늘어나게 되며 Polling 주기가 짧아질수록 쿼리를 사용할 경우 DB의 부하도 같이 늘어나게 된다[12]. Socket.IO는 브라우저 특성 상관없이 일관된 푸쉬 메시지를 보낼 수

있다.

2.3 PM2 프로세스 관리자

Node.js는 커맨드 라인명령어로 실행된다. 여러 종류의 프로세스를 관리하기 위해서는 관리자가 추가로 필요하다. PM2는 프로세스가 계속 실행할 수 있도록 도와주는 역할을 하고 프로세스의 종료 시 다시 실행 시켜 주는 역할을 한다[9]. 보통 Node.js 의 프로세스 개수는 CPU의 코어 개수와 동일하게 본다.

3. 프레임워크 설계 및 구현

3.1 아마존 서버 구축

본 연구의 프레임워크 서버를 구축하기 위해서 퍼블릭 클라우드인 아마존 웹 서비스 AWS를 이용하였다. 기존 게임회사는 IDC를 이용하여 서버를 유지 서버 임대료를 지불 하는 형식이었지만. AWS 의 경우 사용한 시간만큼만 비용을 지불한다. 그리고 클라우드에서의 가상화로 서버의 구축과 제거를 쉽게 할 수 있다. 사용자가 증가하는 시간에는 서버를 쉴게 증설하고 사용하지 않는 서버는 제거하여 관리 할 수 있다. 그리고 서버를 별도로 스냅샷 복사 생성하여 스케일 업 할 수 있고 따로 증설 되는 형태로 서버를 구성하지 않았을 때는 서버 자체의 스펙을 업 시킬 수 있다[4].

3.1.1 아마존 리눅스 구축

AWS에서 지원하는 서버 중 EC2 인스턴스를 이용하여 아마존 리눅스 서버를 구축하였다. EC2 인스턴스는 클라우드 환경에서 컴퓨팅 파워의 규모를 자유자재로 변경할 수 있는 웹 서비스이다.

본 논문에서는 Putty 접속기로 접속하여 작업을 하고 FileZila를 통해 파일의 이동과 권한 관리 프로세스 실행 작업을 하였다. AWS에서 발급하는 PEM Key를 통하여서만 접속을 할 수 있다.

3.1.2 Mysql 5.7 구축

관계형 DB를 구축하기 위하여 AWS에서 제공해주는 서비스인 RDS를 사용하였다. Node.js 는 일반적으로 MongoDB 나 Redis 와 같은 NoSQL DB와 같이 구현을 많이 하지만, 본 연구에서는 데이터의 누적과 보존 유지를 위하여 MySQL으로 DB를 구축하였다[10]. 최근 JSON 타입을 많이 쓰게 되면서 Mysql 5.7.8 버전 부터는 JSON 타입을 지원하며 모든 데이터 타입에 대한 저장이 가능하고 유효성 체크도 가능하게 되었다. 접속기는 가벼운 HeidiSQL을 이용하여 관리하였고 로그 서버는 구축을 따로 하지 않고 단일 스키마에 테이블 구성으로 구축하여 DB를 생성 하였다.

3.2 서버 구현

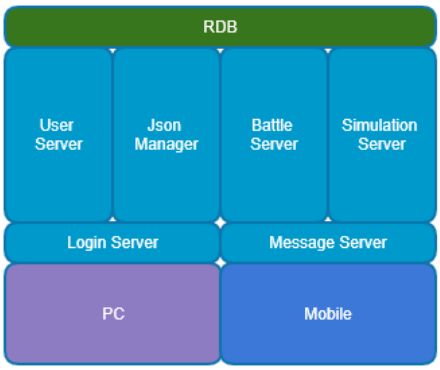
3.2.1 서버 모듈

[Table 1] Use Node.js Module

모듈	기능
well-rng	확률 분포
socket.io	소켓 통신
express	Web 서버
thenby	배열 정렬
mysql	DB 접속기

프레임워크는 [Table 1] 과 같은 모듈을 사용하여 구현하였다. Node.js에서 사용할 수 있는 모듈 중 Well-rng 는 일반 랜덤 함수보다 유용한 난수 생성 알고리즘이다. 서버에서 일어나는 모든 확률 변수들은 이 Well 512 기반의 난수 생성 모듈을 사용하도록 되어있다. thenby 모듈은 배열 sort 함수를 포함하는 모듈이다. 배열을 변경해야 할 경우에 서버에서 이 모듈에 접근한다. mysql 모듈의 경우 DB과 관련된 작업을 도와준다. 접속 연결 유지를 하기 위해 DB Pool을 관리 할 수 있다.

3.2.2 서버 구성도

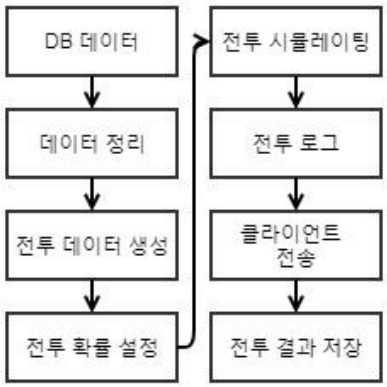


[Fig. 2] Server Configuration Diagram

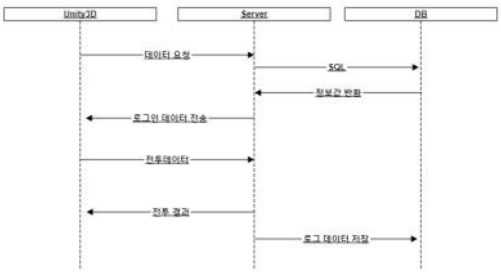
이 프레임워크에서 서버는 [Fig. 2] 와 같은 구조로 설계 되었다. AWS에서는 가상화로 되어있어 서버의 추가 삭제가 편하고 서버 군의 분류를 쉽게 할 수 있다. [Fig. 2] 와 같이 서버는 크게 전투 서버와 유저의 정보를 가져 올수 있는 정보 서버로 나누었다. PC 게임을 제작 할 경우 사용자는 PC 로그인 서버로 접속 한 후 해당 연결되는 전투 서버와 정보 서버를 통하여 전투 시뮬레이션을 할 수 있다.

3.2.3 전투 서버

전투 서버의 흐름은 [Fig. 3] 과 같이 이루어져 있다. 클라이언트에서 JSON 형태로 전달된 데이터를 가지고 전투 서버에서는 전투 시뮬레이션을 한다. 그리고 기록된 전투 로그는 DB에 저장 된다. JSON 자체는 명령어 형식으로 전달되어 서버간 통신을 하고 클라이언트에 해당 결과 값을 반환하여 전투 결과 상황을 보여준다. 이러한 경우 클라이언트에서 전투 결과를 전달하지 않기 때문에 클라이언트의 메모리 해킹에 대해서도 안정성을 보장받을 수 있다.



[Fig. 3] Server flow chart



[Fig. 4] Communication flow chart

3.2.4 정보 서버

정보 서버는 [Fig. 4] 과 같은 통신 흐름도를 갖는다. 전투 정보 데이터를 DB에서 전달 받아 올 수 있는 중간 서버로 게임에 필요한 데이터들을 전달 받을 수 있다. 통신 방법은 다른 서버와 같은 방식으로 JSON 형태의 명령어를 전달받아 서버에서 직접 쿼리 문을 생성한다[11]. 이미 정해진 DB 함수들을 통하여 보다 빠르게 전달 받을 수 있고, 비동기 서버 특성상 많은 사용자가 접속하여도 문제가 없다. [Fig. 4] 와 같이 DB에 따로 추가로 접속하지 않고 곧바로 전투 정보를 되돌려 줄 수 있다.

3.3 클라이언트 구현

하나의 소스를 가지고 멀티 플랫폼으로 배포를

하기 위하여 지금 가장 많이 사용되어지고 있는 Unity3D 엔진을 사용하였다. 개발 버전은 5.6.1을 사용하였다[7].

3.2.1 Unity3D 클라이언트

로그인 Scene 에서는 로그인 서버와 접속하고 다른 서버들의 데이터 정보들을 받아와서 기초 세팅을 해준다. 전투 Scene 에서는 받은 데이터를 기반으로 시뮬레이션 서버에 데이터를 전송하여 결과 값을 받아 올 수 있는 형태로 설계 하였다.

3.2.2 Socket.io 클라이언트 구현

최근 게임에서는 실시간으로 데이터를 처리하기 위해 Websocket을 사용한다. socket.io는 Event 기반으로 통신 처리하기 때문에 Event 등록과 Callback 함수 구현 형태로 제작한다[8].

[Table 2] JSON Configuration Diagram

Key	Value
Unit_id	BigInt
name	String
Hp	int
Attack	int
Accuracy	float

3.2.3 JSON Parser

Unity3D 5.3 버전 이후에서는 JSON API가 추가 되어 있다. iOS/IL2CPP 호환이 되지 않는 것들이 있어서 Unity3D에서는 이러한 문제점들을 파악하고 JSON 라이브러리를 제공한다. 정보서버와 전투서버에서 받은 JSON 파일을 클라이언트에 표시 할 수 있게 구현하였다. [Table 2] 와 같이 Key 와 Vaule 값을 가지고 있기 때문에 원하는 값을 다양한 언어의 형태로 가져 올 수 있다. Unity3D 의 경우 C# 으로 되어 있기 때문에 Map 형태로 값을 활용 할 수 있다.

4. 성능 분석

기존 전투 시뮬레이션 네트워크의 경우는 C++, MFC, DB를 이용하여 모의 모형을 구현하였다. 기존 DB의 경우 시뮬레이션 도중의 Log를 저장하는 형태로 사용하였으며 최대 200회 반복 실행을 하여 평균 추청 치에서 2배 표준 오차 범위 내에서 오차 범위를 측정하였다[5].

기존 연구에서는 시뮬레이션의 반복 실행을 100~200번의 시뮬레이션 밖에 하지 못하여 예상 밖의 범위에서는 대응을 할 수 없다. 그리고 해당 확률에 대해서는 정해진 게임성의 목적으로 확률이 수렴하게 되어야 하는데 표본이 작으면 변동성과 오차범위가 커지게 된다. 이것을 해결하기 위하여 시뮬레이션 범위를 최대 10,000,000회 까지 가능하게 하고 최대 실행 시간은 60분미만으로 할 수 있게 하였다. 그리고 해당 로그 및 승리 기록은 데이터로 남게 하였다.

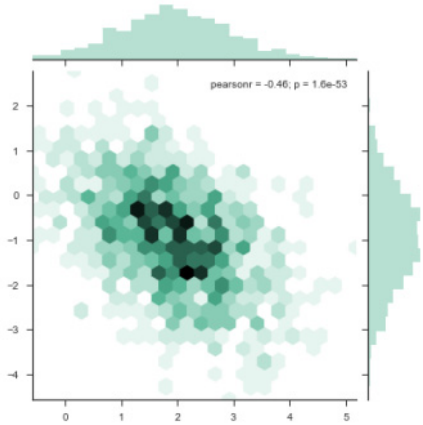
4.1 성능 분석 환경

본 벤치마크 테스트를 위하여 [Table 3]와 같은 성능의 마이크로 서비스 테스트용인 M4 모델을 사용하여 성능 환경을 구축하였다. AWS에서 제공해주는 M4 모델의 경우 6가지가 있는데 가장 컴퓨팅 스펙이 적은 모델로 선택하였다[6].

[Table 3] Test System Environment

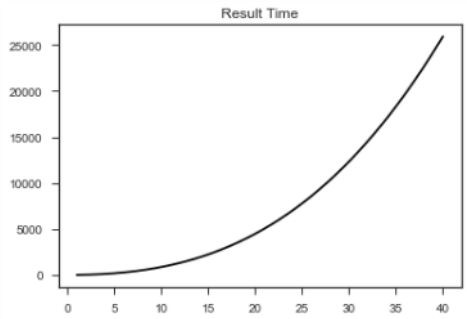
	AWS EC2
모델	m4.large
vCPU	2
메모리	8 GiB
SSD	EBS 전용
EBS	450

4.2 성능 분석 결과



[Fig. 5] Battle Result Diagram

시뮬레이션 결과는 X축의 경우 아군의 승리 점수이고 Y축의 경우는 적군의 승리 점수이며 [Fig. 5]와 같이 전투의 결과는 상단과 우측의 밀도 분포를 보면 정규 분포 형태를 보인다. 기존 연구의 확률 형 전투 모델에서 크게 벗어나지 않고 있는 것으로 보아 실제 사용 가능한 안전성을 가지고 있다.



[Fig. 6] Result Time Diagram

테스트 전투 시뮬레이션의 10만회 반복에 약 27초의 연산 시간이 소요되었다. 로그의 기록을 남길 경우, 파일I/O와 메모리의 사용량이 증가 하지만 기본 시뮬레이션의 결과값 만을 출력할 때의 소요 시간 대비 전투횟수는 증가하게 되는데, 시간당 전투횟수의 증가 양상은 [Fig. 6]과 같은 형태를 가지게 된다. 보다 더 상위 스펙의 서버를 사용하면 시뮬레이션 소요 시간이 더 줄어들 수 있다.

5. 결 론

본 프레임워크를 이용하면 서버 의존성이 높은 전투 시뮬레이션 게임을 보다 쉽게 제작하고 배포함으로써 개발기간의 단축 및 게임 밸런싱을 효과적으로 할 수 있다. AWS 서비스를 이용하게 되면서 사용자가 늘어나면 컴퓨팅 파워 및 게임 서버를 쉽게 늘릴 수 있는 장점을 가질 수 있다. 따라서 시스템의 운영 및 구입비용을 절약할 수 있다. 여러 플랫폼을 이용가능하기 때문에 단일 플랫폼을 선정하거나 멀티 플랫폼 선정 시, 개발에 따른 추가 작업이 필요하지 않아 플랫폼에 독립적인 개발을 할 수 있다. 특히 Socket.io 과 Node.js 의 결합으로 비동기 처리를 이용하여 많은 사용자가 접속하여도 빠른 응답 속도를 보장 할 수 있고 구축하기 힘든 실시간 TCP/IP 형태의 서버를 쉽게 Javascript 로 제작함으로써 개발자 확보 및 충원에 부담을 줄 일 수 있다.

앞으로의 연구 방향은 관계형 DB와 NoSQL 을 결합한 신규 서버를 제작하여 DB의 부하를 줄 일 수 있는 버퍼형태의 DB 서버를 구현하고, 별도의 로그 서버를 구축함으로써, 실시간으로 전체 게임의 상태를 알 수 있는 시스템을 설계 하는 것이다.

REFERENCES

[1] Lim Jung-yul, Park Il-gyu, Jae-yong Chung,, “Distributed Game Server Technology Trends”, Korea Game Society, 20 Volume 4, p93, 2005. 08. 05

[2] Chae Won Chae, Park Chan Woo, Choi Wan, Ahn Se Young, Noh Byeongseok, Lee Jun Woo, "HTML5 and Framework Trends for Mobile Web Apps", Volume 27, Issue 3, p92, June 15, 2012.

[3] Sun Young Bum, An Mi Sun, Kim Tae

Yong, Lee Won Hyung, "Design of Hybrid Client Server System for Online Game", Journal of Korea Computer Game Society, pp44-50, 2002.10,
[4] Yong-Bin Kim, Dong-Kyu Shin, Dong-Il Shin, "Implementation and Performance Analysis of Simultaneous Wired and Wireless Game Server", Proceedings of the Korean Information Science Society 30 (2), pp694-696, 2003.10
[5] Min Hyun Joon and Hong Yoon Ki, "Simulation of Combat Network Following Probability Process," Journal of the Korea Society for Simulation, 19 (1), pp113-123, 2010.3,
[6] https://aws.amazon.com/ko/ec2/instance-types/
[7] https://docs.unity3d.com/Manual/index.html
[8] https://socket.io/docs/
[9] http://pm2.keymetrics.io
[10] Lee, Myung - Sun Lee, Hyung - Yong Oh, Min Byung Won, (2011). Improved mobile web usability optimized for mobile cloud computing. The Journal of the Korea Contents Association, 11 (9), 85-95.
[11] Oh, Jin - Soo, Song Chang - Ki. (2012). Improving performance by transforming XML data into JSON in mobile applications. Korean Information Science Society Proceedings, 39 (1D), 129-131.
[12] Lee, Joon - Gyu, Lim Kyung - Soo, In Sun Shin. (2003). Design and Implementation of Priority Transaction Layer on Java - based WAP. Journal of the Institute of Information Science and Technology, 30 (2), 244-251.

권 순 정(Kwon, Soon Jung)

약 력 : 1988 동국대학교 전자계산학과 석사
2009 상명대학교 게임학과 박사 수료
1991-1997 KB데이터시스템 SI 사업부
1997-2007 영진전문대학교 컴퓨터정보기술 계열(컴퓨터게임 전공) 조교수
2007- 서강대학교 평생교육원 MTEC (서강대학교 게임교육원) 교수

관심분야 : 게임기획, 게임프로그래밍, 기능성게임