Journal of Korea Game Society 2015 Apr; 15(2): 83-94
http://dx.doi.org/10.7583/JKGS.2015.15.2.83

게임 프로그래밍

온라인 게임 서버의 데이터연동 방식에 대한 연구

강민석*, 김경식**
호서대대학원 컴퓨터공학과*, 호서대학교 게임학과**
jusin26@naver.com, kskim@hoseo.edu

A Study on the Data Linkage Method of Online Game Server

Min-Seok Kang*, KyungSik Kim**
Dept. of Computer, Graduate School of Hoseo University*
Dept. of Game Development, Hoseo University**

요 약

온라인 게임 개발을 위해서는 서버 개발 기술이 중요하다. 왜냐하면 사용자들이 게임을 진행하는 클라이언트가 서버에 종속적으로 구현되기 때문이다. 본 연구는 서버 개발 기술 중 서버가 데이터를 데이터베이스에 저장하고 불러오는 데이터연동 방식을 연구하였다. 서버의 데이터 연동은 장소에 관계없이 게임을 이어서 진행할 수 있는 온라인게임의 특성 때문에 매우 중요한 사항이다. 본 논문에서는 여러 데이터연동 방식을 정의와 함께 제안하며 분류하였다. 제안된 방식들을 사례를 통해 평가하였다. 본 연구를 통해 향후 온라인 게임 서버의 개발 시에 게임의 특징에 맞는 데이터연동 방식을 선택할 수 있을 것으로 기대된다.

ABSTRACT

Methodology of server development is so important for the development of online game because the client where the users play the online game depends on the server. In this research, we have studied data linkage methods of the online game server which permits the server to save and retrieve the user data on the database. The data linkage method of the server is critical to make the users continue their games anywhere online. In this paper, several data linkage methods have been proposed and classified with their definitions. The proposed methods have been evaluated with the experimental data. It will contribute for the developer of online game server to choose a good data linkage method for the features of the game.

Keywords : Online Game(온라인 게임), Game Server(게임 서버), Game Database (게임 데이터베이스), Server Structure(서버 구조), Data Linkage(데이터 연동)

Received: Mar. 11, 2015 Accepted: Mar. 31, 2015
Corresponding Author: KyungSik Kim(Hoseo University)
E-mail: kskim@hoseo.edu

ISSN: 1598-4540 / eISSN: 2287-8211

1. 서 론

한국 게임 산업은 최근 각종 규제들에 지속[1]되어 우려를 낳고 있지만 나날이 발전하고 있다. 이 중 2012년 전체 매출의 69.7%를 차지하는 온라인 게임이 가장 중요한 분야이다[2].

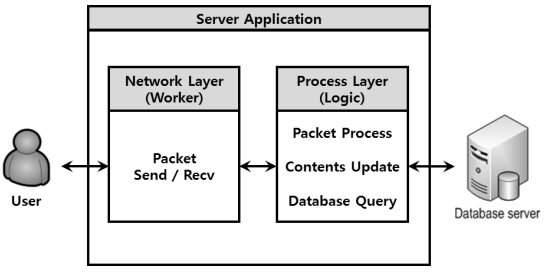
이러한 온라인 게임 산업에서 서버 개발 기술은 다른 게임 산업과 구분 짓는 필수적인 기술이다[3]. 왜냐하면 사용자들이 게임을 진행하는 클라이언트가 서버에 종속되어 구동되기 때문이다. 서버 개발 기술에 대한 연구는 서버 구조에 대한 연구와 네트워크 모델에 대한 연구 등이 있으며 본 연구는 서버 구조에 대해 논한다. 특히 그동안 연구가 미흡하였던 서버의 데이터연동 방식에 대한 연구를 진행하였다. 서버의 데이터연동은 장소에 관계없이 게임을 이어서 진행할 수 있는 온라인게임의 특성 때문에 매우 중요한 사항이다. 하지만 그동안 서버의 데이터연동에 대한 연구는 많은 시간과 비용이 소요되는 서버 연구의 어려움 때문에 연구가 미흡하였다.

본 연구는 단일 서버 환경에서 서버가 데이터를 데이터베이스에 저장하는 여러 가지 방식에 대해 제안하고 평가하는 것을 목표로 한다. 이를 위해 먼저 기존에 연구된 서버 계층별 구현 방식에 따른 4가지 모델을 살펴보았다. 이후 서버의 데이터 연동 계층에 대해 정의하고 4가지 기준에 따라 각각 2종류의 세부 구현 방식을 제시하였다. 그리고 제시된 방식들에 대해 성능 평가를 진행하였다. 제시된 성능 평가 결과를 토대로 구현하고자 하는 게임의 특성에 맞는 데이터연동 방식을 선택할 수 있을 것으로 기대된다.

2. 관련 연구

2.1 게임 서버의 계층과 4가지 모델

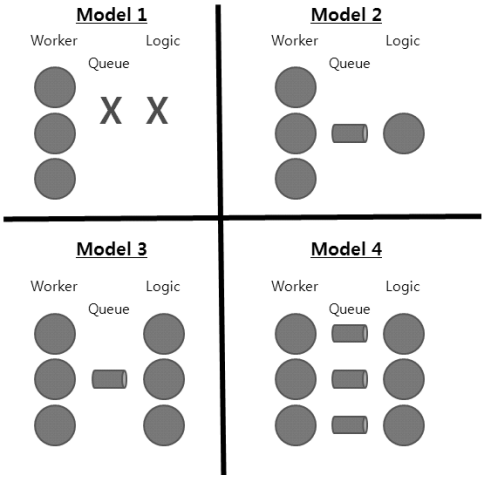
일반적인 게임 서버의 구조는 [Fig. 1]와 같이 네트워크 계층과 프로세스 계층으로 나눌 수 있다.



[Fig. 1] Online Game Server's Layer

네트워크 계층은 소켓을 사용하여 패킷을 송수신하고 이벤트를 감지하는 것을 담당하게 된다. 윈도우용 서버의 경우 주로 IOCP(Input/Output Completion Ports, 마이크로소프트가 윈도우즈 운영체제에서 제공하는 고성능 IO모델)[4,5]를 사용하게 개발이 된다. 프로세스 계층은 서버 어플리케이션이 담당해야 될 작업을 수행하는 계층이다. 이 계층은 콘텐츠 처리 계층 또는 논리 계층이라는 말로 표현되기도 한다.

서버는 네트워크 계층과 프로세스 계층의 연동 방식에 따라 주로 4가지 형태의 모델[6]로 분류된다. [Fig. 2]의 Worker 는 네트워크 계층의 구조를 말하고 Logic 은 프로세스 계층의 구조를 말하며 Queue 는 계층 간 연동을 위해 사용되는 쓰레드 안전(thread-safe)한 버퍼를 말한다.



[Fig. 2] Online Game Server's 4 Models[6]

Model 1은 네트워크 계층과 프로세스 계층이 결합된 형태를 말한다. 네트워크 계층에서 어떤 요구를 전달 받으면 다른 가공을 거치지 않고 곧바로 처리해버리는 것이다. 이 방식의 장점은 할당된 쓰레드의 숫자에 따라 서버의 성능을 극대화할 수 있다는 점이다. 하지만 프로세스 계층이 다중쓰레드로 구성되어 있기 때문에 개발 난이도가 상승하며 교착 상태(dead-lock) 등의 동기화 문제가 생길 가능성이 높다.

Model 2는 네트워크 계층에서 받은 요구들을 하나의 버퍼에 누적시킨 후 프로세스 계층에서 처리하도록 하며 프로세스 계층을 단일쓰레드로 구성하는 방법이다. 프로세스 계층이 단일쓰레드로 구성되어 있어 구조가 단순해지고 개발 난이도가 낮은 장점이 있다. 하지만 단일쓰레드로 구성된 프로세스 계층의 부하가 집중되는 문제가 있고 때문에 성능이 감소되는 단점이 있다.

Model 3는 Model 2의 단점을 개선하여 프로세스 계층을 다중쓰레드로 구성한 모델이다. 이 모델의 장점은 다중쓰레드로 프로세스 계층의 부하를 분산하여 서버의 성능을 극대화할 수 있는 점이다. 하지만 Model 1에서와 같이 프로세스 계층의 다중쓰레드는 개발난이도를 상승시키고 유지보수를 어렵게 하는 단점이 있다. 또한 네트워크 계층의 명령을 누적하는 버퍼에 너무 많은 쓰레드가 접근하여 부하를 일으키는 경우가 생긴다.

Model 4는 Model 3의 단점인 버퍼의 부하를 줄일 수 있도록 버퍼를 다중 구성한 모델이다. 구조가 단순하여 성능이 가장 좋은 Model 1에 가장 근접한 성능을 낼 수 있다. 하지만 다중 구성된 버퍼 내부에서 명령간의 수행 순서가 보장되지 않아 개발 난이도가 4가지 모델 중 가장 높다.

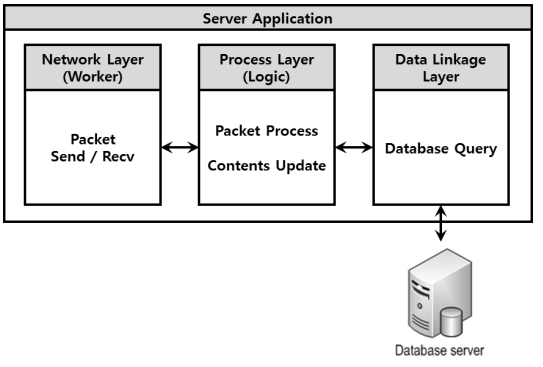
2.2 그 외 관련 연구

서버 구조에 대한 연구 중 분산 게임 서버 구조에 대한 연구는 ETRI 네트워크가상연구팀이 발표한 분산 게임 서버 기술 동향[7]이 가장 대표적인 연구라 할 수 있다. 다양한 형태의 분산 게임 서버를 제시하고 분류하였고 향후 전망에 대해 소개하였다. 분산 게임 서버 구성 시 효율적으로 확장성을 높이기 위한 연구[8,9]가 있다. 그리고 맵 분할 방식에 따른 효과적인 서버 구조에 대한 연구[10,11,12]가 있다. 그 외에 온라인 게임의 트래픽을 조사한 연구[13,14]가 있다. TCP 와 UDP 같은 프로토콜 측면과 RTS, FPS, MMORPG의 네트워크 사용량에 대해 분석하여 장르별로 비교하는 연구를 하였다.

3. 본 론

3.1 데이터연동 계층의 정의

2장에 언급된 프로세스 계층은 패킷 처리와 게임 콘텐츠 갱신, DB 질의 등 여러 가지를 담당하고 있는데 이 중 DB 질의를 다른 계층으로 나누어 [Fig. 3]와 같이 세분화 할 수 있다.

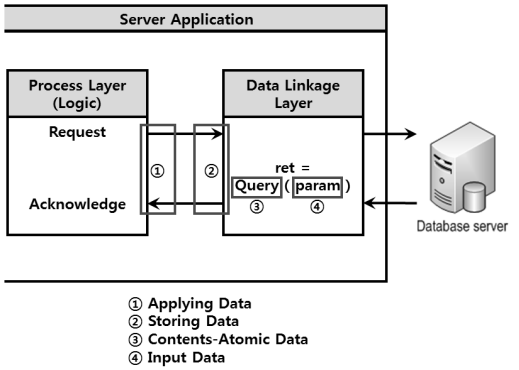


[Fig. 3] Online Game Server's 3 Layer

데이터연동 계층은 게임 내 각종 데이터를 데이터베이스와 연동하는 계층이다. 이 계층은 데이터베이스로부터 데이터를 불러오고 데이터를 저장하는 작업을 담당하며 데이터베이스가 담당하는 데이터의 종류와 양[15]에 따라 달라지지만 주로 다중쓰레드로 처리된다.

3.2 데이터연동 계층 세부 구현에 따른 분류

데이터연동 계층을 구현하는 방식들은 [Fig. 4]와 같이 데이터 적용 방식과 데이터 저장 시기, 콘텐츠-원자성(atomicity) 보장 여부, 입력 방식에 따라 분류 가능하다.



[Fig. 4] Data Linkage Layer's Flow and Categorization

데이터연동 계층은 [Table 1]과 같이 4가지 기준에 따라 각각 2가지 방식으로 구현이 가능하며 서로 결합되어 구현될 수도 있다.

[Table 1] Data Linkage Layer's Implementation Methods

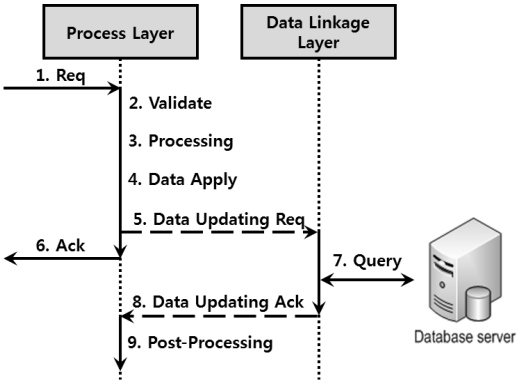
	method	detail
applying data	data pre-apply	-
	data post-apply	sync
		async
storing data	saving after updating	-
	saving after cashing	time
		event
contents-atomic data	contents-atomic guarantee	-
	contents-atomic no guarantee	-
input data	absolute-value input	-
	relative-value input	-

3.2.1 데이터 적용 방식에 따른 분류

데이터를 서버에 적용하는 방식은 데이터연동 계층과 실제 데이터를 소유한 데이터베이스의 관계로 볼 수 있다. 데이터 적용 방식은 데이터 적용시점에 따라 데이터를 선적용하는 방식과 후적용하는 방식으로 나뉜다.

가. 데이터를 서버에 선적용하는 방식

데이터를 서버에 선적용하는 방식은 [Fig. 5]과 같이 총 9가지 순서로 수행되고 크게 3가지 단계로 구성된다. 첫 번째 단계에서는 요청을 받은 프로세스 계층에서 서버 어플리케이션의 메모리에 데이터를 반영한 후 데이터연동계층에서 데이터베이스로 질의를 요청한다. 두 번째 단계에서는 데이터를 데이터베이스에게 질의한 결과가 성공한 것으로 간주하고 이미 반영된 정보를 바탕으로 멈춤 없이 계속 서비스를 진행한다. 이 때 데이터베이스에서 특수한 경우를 제외하고 실패 응답이 발생하지 않도록 서버에서는 첫 번째 단계에서 유효성 검사를 충분히 한 후 질의한다. 세 번째 단계는 데이터베이스의 응답을 확인하는 단계로 만약 데이터베이스에서 뒤늦게 실패한 것으로 응답이 오는 경우 서버는 이미 서버에 적용된 내용을 되돌린다.



[Fig. 5] Data pre-apply Method Flow

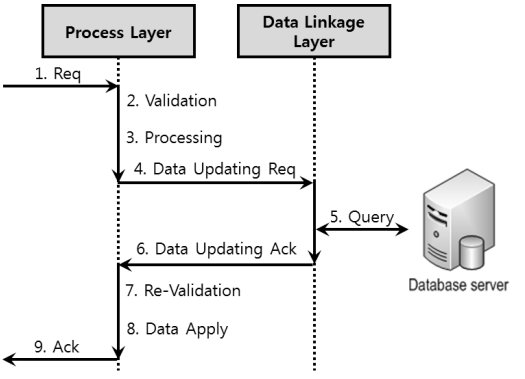
나. 데이터를 서버에 후적용하는 방식

데이터를 서버에 후적용하는 방식은 데이터연동 계층에서 데이터 처리 후에 프로세스 계층에서 서버 메모리에 반영하는 방식을 말한다. 이 때 프로세스 계층에서는 사용자 요청의 유효성에 대해서만 검증한 후 데이터연동 계층으로 요청을 위임하고 데이터연동 계층에서의 응답을 기다린다. 데이터연동 계층에서는 데이터베이스로 질의를 요청하고 응답을 받고 프로세스 계층으로 받은 응답을 전달하게 된다. 프로세스 계층에서는 전달받은 응답을 메모리에 반영하고 사용자 요청에 대한 응답을 하게 된다.

데이터 후적용 방식은 데이터연동 계층에서 데이터에 대한 요청과 응답을 처리하기 때문에 계층에 따른 모듈화가 잘 되어 개발이 쉬운 장점이 있다. 또한 선적용 방식과 다르게 데이터베이스에서 실패 응답이 오더라도 동기화에 따른 이슈가 없기 때문에 쉽게 처리가 가능하다. 이에 따라 데이터베이스에서 능동적으로 유효성 및 무결성에 대해 검증하고 처리할 수 있어 데이터베이스의 활용도가 높다. 데이터 후적용 방식의 단점은 원격지에 있는 데이터베이스로부터 응답을 받아온 후에 사용자 요청에 대한 응답을 하기 때문에 반응속도가 느리다는 점이 있다. 또 프로세스 계층의 응답 대기 방식이 동기 방식일 경우 원격지로부터 응답이 올 때까지 프로세스 계층이 멈추게 되고 이는 곧 서버의 부하로 이어지게 되는 문제가 있다. 프로세스 계층의 부하에 취약한 서버 어플리케이션의 Model 2를 사용할 경우에는 동기 방식을 사용하여서는 안된다.

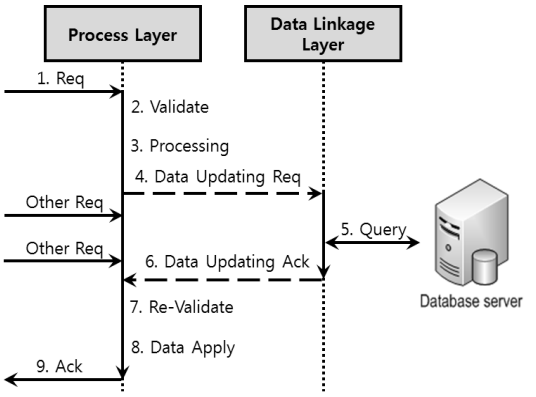
데이터 후적용 방식은 프로세스 계층이 데이터연동 계층으로부터의 응답을 기다리고 처리하는 방식에 따라 동기 방식과 비동기 방식으로 다시 나눌 수 있다.

동기 방식은 데이터연동 계층에 요청 후 응답이 올 때까지 해당 프로세스 계층이 대기를 하는 방식이다[Fig. 6]. “사용자 요청-유효성검사-데이터연동 요청-데이터베이스 질의-데이터연동 응답-데이터 후적용-사용자 응답“으로 이어지는 일련의 흐름을 일관된 하나의 흐름으로 볼 수 있어 개발 쉽고 직관적인 장점이 있다. 이 장점은 서버 어플리케이션 Model 1 방식과 결합되면 완전히 하나의 로직으로 묶일 수 있어 더욱 극대화 될 수 있다. 하지만 데이터베이스 질의 시 응답이 늦어질 경우 프로세스 계층에 부하를 주게 되어 서버 전체의 응답속도가 늦어지는 문제가 있을 수 있다. 서버 어플리케이션의 프로세스 계층이 충분한 숫자의 다중쓰레드로 구성되어 있어야 이 문제를 완화할 수 있다.



[Fig. 6] Data post-apply Method Flow (Sync)

비동기 방식은 프로세스 계층이 데이터연동 계층으로 요청 후에 응답을 기다리지 않고 다른 사용자 요청을 계속 처리하는 방식을 말한다[Fig. 7]. 이 때 데이터연동 계층으로부터 응답을 기다리는 사용자의 다른 요청은 응답이 올 때까지 개별 버퍼에 저장 해두어야 한다. 데이터연동 계층으로부터 응답 시까지 대기하지 않고 다른 사용자의 요청을 처리하기 때문에 응답 속도가 빨라지는 장점이 있다. 하지만 일련의 흐름을 하나의 로직으로 처리할 수 없어 데이터 후처리방식의 장점인 구조의 단순화가 줄어들게 되는 단점이 있다.



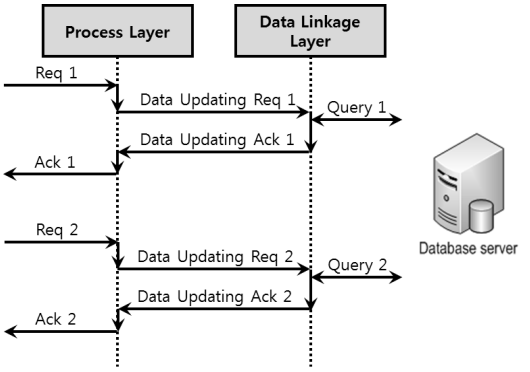
[Fig. 7] Data post-apply Method Flow (Async)

3.2.2 데이터 저장 시기에 따른 분류

서버가 사용자로부터 데이터의 갱신 요청을 받았을 때 데이터를 실제 데이터베이스에 저장하는 시기에 따라 서버 구조를 분류할 수 있다. 데이터를 갱신 즉시 데이터베이스에 저장하는 방식과 갱신들을 서버에 캐싱 하였다가 데이터베이스에 저장하는 방식이 있다. 서버에 캐싱 후 데이터베이스에 저장하는 방식은 일정 시간마다 데이터베이스에 저장하는 방식과 사용자 상태 전환과 같은 중요한 이벤트 때에만 저장하는 방식이 있다. 즉시 저장하는 방식은 데이터 적용 방식으로 선적용과 후적용 중 선택이 가능하지만 캐싱 후 저장하는 방식은 캐싱 과정에서 데이터 선적용 방식이 일부 포함된다. 캐싱 시에는 무조건 데이터 선적용 방식이 적용되고 후에 캐싱된 데이터를 실제 데이터베이스에 저장 시에는 데이터 선적용과 데이터 후적용 중 택일 할 수 있는 것이다.

가. 갱신 즉시 저장 방식

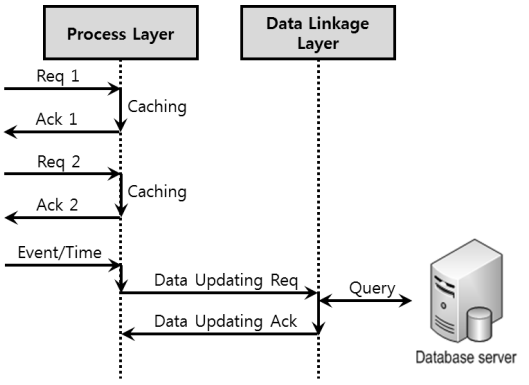
갱신 즉시 저장 방식은 프로세스 계층에서 데이터의 갱신이 필요한 경우 즉시 데이터연동 계층으로 요청을 하여 데이터베이스에 질의하여 갱신정보를 저장하는 방식이다[Fig. 8]. 이 경우 데이터베이스에 질의하는 횟수가 많아져 데이터베이스 성능이 떨어진다. 만약 데이터 적용 방식을 후적용 방식으로 할 경우 데이터베이스에 요청하는 질의 횟수가 많아진 만큼 서버의 성능에도 좋지 않은 영향을 준다. 하지만 서버에서 별도로 데이터 사본을 가지고 있지 않아 데이터베이스와의 동기화 이슈가 작고 서버의 안정성이 증가하는 장점이 있다. 또한 서버 예외상황 시 데이터가 갱신되지 못하고 누락되거나 소실되는 경우가 적다. 데이터 갱신과 데이터베이스의 저장 간에 시간 차이가 크지 않기 때문이다.



[Fig. 8] Saving after Updating Method Flow

나. 서버에 캐싱 후 데이터베이스에 저장하는 방식

캐싱 후 저장 방식은 프로세스 계층에서 데이터의 갱신이 필요한 경우 데이터베이스에 저장하지 않고 서버에서만 데이터를 누적하여 사용하다가 후에 데이터베이스에 저장하는 방식이다[Fig. 9]. 데이터베이스에 저장하지 않고 데이터를 누적하여 사용한다는 것은 데이터를 캐싱한다는 의미로 서버 내부의 데이터 객체에 데이터 적용 후에 대기 없이 다음 동작을 진행하는 것이다. 후에 데이터베이스에 저장한다는 것은 일정 시간이 경과한 후나 특정 이벤트 시에 데이터베이스에 저장되지 않고 누적되어 있던 데이터들을 저장한다는 것을 뜻한다.



[Fig. 9] Saving after Caching Method Flow

캐싱 후 저장 방식은 데이터를 얼마나 오래, 몇 회나 누적하느냐에 따라서 성능과 안정성에 큰 차이를 보이게 된다. 성능적인 측면에서는 반대로 오래 누적하면 할수록 데이터베이스 질의 횟수가 감소하여 성능에 증가하게 된다. 하지만 안정성 측면에서는 오래 누적하면 할수록 데이터 소실의 위험이 커져 문제가 된다. 오래 누적할수록 데이터베이스에 실제로 저장된 데이터와 차이가 커지게 되어 크래시와 같은 서버의 예외상황에서 소실되는 데이터가 많아지기 때문이다.

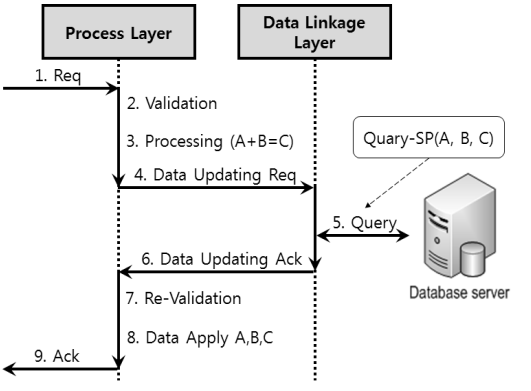
일반적으로 주기적인 시간에 따라 저장하는 방식과 사용자 상태 변화 같은 중요한 이벤트 시에 저장하는 방식을 모두 사용하여 캐싱된 데이터를 데이터베이스에 저장하게 된다. 두 방식이 서로 배타적이지 않아 동시에 적용이 가능하기 때문이다.

3.2.3 콘텐츠-원자성 보장 여부에 따른 분류

데이터를 갱신할 때 콘텐츠에 따라 원자성을 가지는 경우가 있다. 원자성이라 함은 어떠한 작업이 실행될 때 언제나 완전하게 진행되어 종료되거나, 그럴 수 없는 경우 실행을 하지 않는 경우를 말한다. 예를 들어 MMORPG에서 한 사용자가 몬스터를 사냥하였다고 가정해보자. 사용자가 몬스터를 사냥한 보상으로 경험치와 아이템을 동시에 획득하게 된다면 이를 원자성을 가진 콘텐츠라고 볼 수 있다. 이 때 경험치와 아이템을 하나의 트랜잭션(transaction)으로 동작시키는 경우 콘텐츠-원자성이 보장된다고 볼 수 있고 개별 동작시키는 경우에는 콘텐츠-원자성이 보장된다고 볼 수 없다. 따라서 콘텐츠-원자성은 하나의 콘텐츠에서 발생한 여러 갱신을 나눌 수 없는 하나의 단위로 보는 것이라 할 수 있다. 콘텐츠-원자성 보장은 서버의 데이터연동 계층에서 데이터베이스로의 질의 명령에서 결정된다.

가. 콘텐츠-원자성 보장 방식

콘텐츠-원자성 보장 방식은 콘텐츠가 원자성을 필요로 할 경우 데이터연동 계층에서 하나의 트랜잭션으로 묶인 한 번의 질의요청만을 하는 방식이다[Fig. 10]. 이 경우 서버나 데이터베이스의 예외상황에서도 정확한 결과를 보장해 줄 수 있어 안정성이 증대된다. 하지만 데이터베이스에서 처리하여야 하는 단일 질의문의 크기가 커져 성능에 좋지 않은 영향을 주며 이는 곧 서버의 부하로 이어진다. 그리고 데이터 단위가 하나로 묶이기 때문에 캐싱이 힘들어져 질의 호출 횟수가 늘어날 수도 있다. 또한 콘텐츠 별로 질의문을 개별 개발하여야 하기 때문에 개발 비용이 크다.

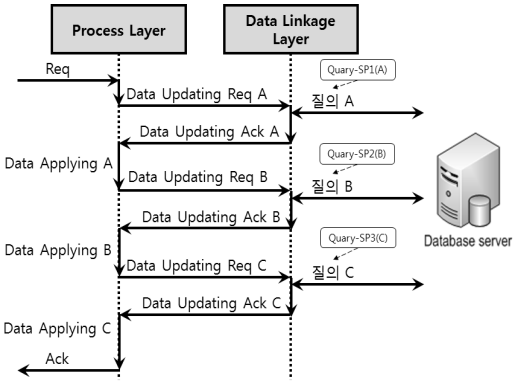


[Fig. 10] Contents-Atomic guarantee Method Flow

나. 콘텐츠-원자성 비보장 방식

콘텐츠-원자성 비보장 방식은 콘텐츠의 원자성 여부와 관계없이 데이터베이스 갱신 측면에서 개별적으로 질의를 요청하는 방식을 말한다[Fig. 11].

데이터베이스의 질의 호출 횟수가 많아지지만 질의문의 크기가 작아져 성능이 좋다. 그리고 콘텐츠의 내용과 무관하게 질의문을 공유하여 사용할 수 있어 개발 비용이 작다. 하지만 서버와 데이터베이스의 예외상황 시에 원자성이 보장되지 않아 콘텐츠 단위의 무결성이 보장되지 않는 단점을 가진다. 예를 들어 FPS 게임에서 대전 횟수와 승패 횟수를 개별 저장할 경우 서버에 예외상황이 발생하면 합산이 맞지 않는 경우가 생길 수 있다는 것이다.



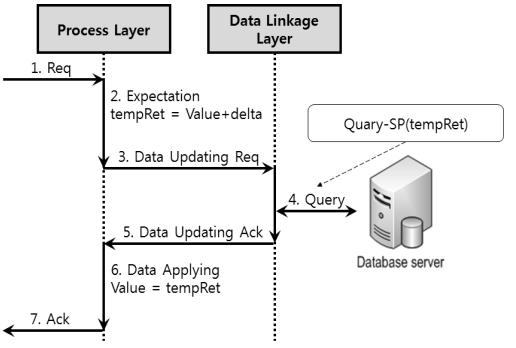
[Fig. 11] Contents-Atomic no guarantee Method Flow

3.2.4 데이터 입력 방식에 따른 분류

데이터를 데이터베이스에 입력할 때 절대값을 입력하여 대입하는 방식과 상대값을 입력하여 누적하는 방식이 있다. 서버에서 데이터베이스의 처리와 무관하게 요청에 대한 최종 결과 값을 이미 알고 있는 경우에만 이러한 선택이 가능하다.

가. 절대값 전달 방식

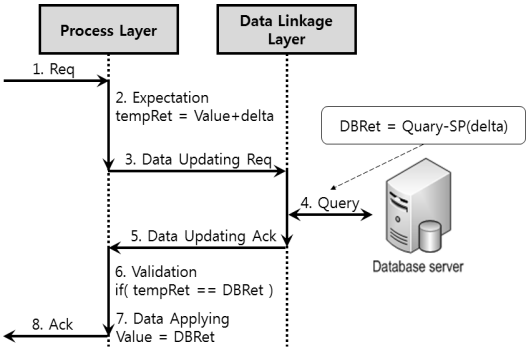
절대값 전달 방식은 갱신을 하기 위한 데이터의 값을 바로 데이터베이스에 전달하여 대입하는 방식이다[Fig. 12]. 구현이 간단하고 데이터베이스에서 할 일이 적어 개발편의성이 높고 성능이 뛰어나다. 하지만 데이터베이스에서의 유효성 검사가 힘들고 버그가 있을 경우 발견이 힘든 점 등이 문제로 안정성이 떨어진다. 또한 이를 악용하여 데이터베이스로 유효하지 않은 값을 전달하는 등의 해킹 행위가 발생할 수 있다.



[Fig. 12] Absolute-Value Input Method Flow

나. 상대값 전달 방식

상대값 전달 방식은 갱신을 하기 위한 데이터의 값을 이전 데이터 값과의 차이를 전달하여 데이터베이스에서 누적 후 최종값을 다시 전달 받는 방식이다[Fig. 13]. 구현이 복잡하여 개발편의성을 떨어뜨리고 데이터베이스의 성능 저하가 조금 있지만 안정성이 높아진다. 서버의 메모리상 데이터와 데이터베이스에 실제 저장된 값의 차이로 인해 아이템 복사 등 많은 문제가 생길 수 있는데 이를 예방할 수 있기 때문이다. 상대값 전달 방식은 유효성 검사를 데이터 갱신 전후로 정밀하게 할 수 있어 이 문제를 예방할 수 있다. 데이터 후적용 방식 및 데이터 즉시 저장 방식과 함께 사용되면 안정성 높은 시스템을 구축할 수 있다.



[Fig. 13] Relative-Value Input Method Flow

3.3 사례연구 및 성능 평가

3.3.1 사례연구

“충무공 해상대전”[16,17]은 온라인 RTS 게임으로 2008년 아산시의 지원 하에 호서대학교-(주)테이란이 개발한 게임이다[Fig. 14]. 2009년 4월 아산시의 ‘성웅 이순신 축제’에서 e-sport 경기를 진행[18]하였으며 이후로도 매년 경기를 진행하고 있다.

[Fig. 14] Chungmukong's Battle on the Sea

3.3.2 데이터 적용 방식 별 시뮬레이션

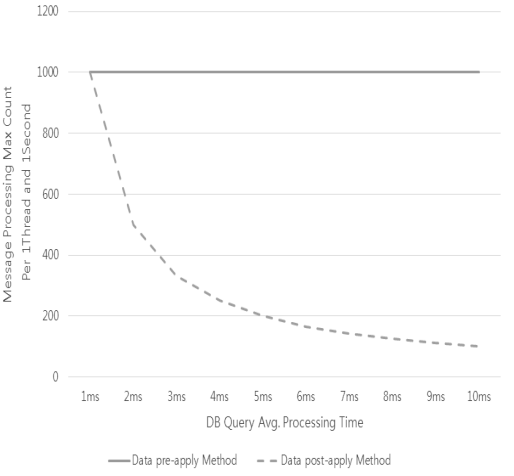
가. 시뮬레이션 환경

데이터 선적용 방식과 데이터 후적용 방식에 대한 성능은 사례인 “충무공 해상대전”에 이 방식들을 각각 적용하여 시뮬레이션함으로써 평가해 볼 수 있다. 시뮬레이션을 위해 DB 질의 수행 속도는 정해진 고정값으로 진행하였다. 또한 DB 질의 수행함수는 블로킹 방식으로 동작하도록 하였다.

시뮬레이션에 사용된 서버의 동작 환경
- CPU: Intel i5 760 2.80 GHz (Quad-Core)
- RAM: 4 GB
- OS: Windows Server 2008 R2 64bit

나. 실험 결과

시뮬레이션 결과는 [Fig. 15]과 같다. 1개 쓰레드 당 메시지 최대 처리 횟수는 서버 어플리케이션이 같은 시간 내에 얼마나 많은 메시지를 처리할 수 있는지를 나타내기 때문에 성능의 척도라 볼 수 있다. 이러한 메시지 최대 처리 횟수가 데이터 선적용 방식의 경우 일정하게 유지되지만 데이터 후적용 방식의 경우에는 DB 질의 수행 속도에 따라 급격히 떨어지는 것을 알 수 있다. 따라서 데이터 후적용 방식의 성능이 데이터 선적용 방식에 비해 좋지 않다는 것을 알 수 있다.



[Fig. 15] Chungmukong's Battle on the Sea

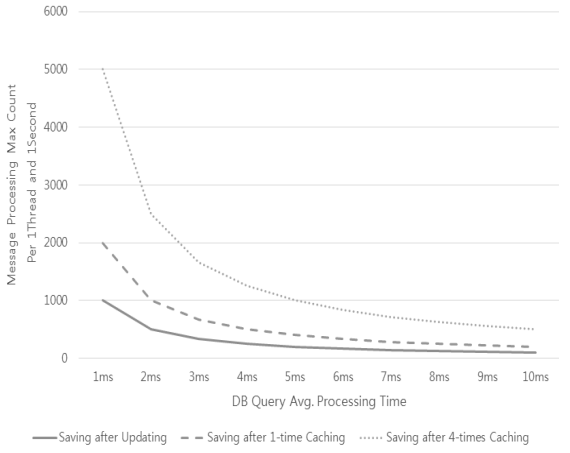
3.3.3 데이터 저장 시기 별 시뮬레이션

가. 시뮬레이션 환경

데이터 저장 시기에 따른 방식들의 평가는 성능과 안정성에 대한 측면으로 나누어 진행하였다. 앞서 데이터 적용 방식에 대한 평가와 마찬가지로 시뮬레이션을 위해 DB 질의 수행 속도는 정해진 고정값으로 진행하였으며 DB 질의 수행함수는 블로킹 방식으로 동작하도록 하였다. 시뮬레이션에 사용된 서버의 동작 환경은 3.3.2의 환경과 같다.

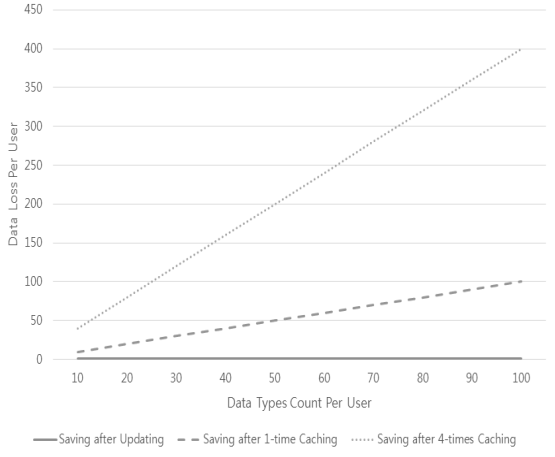
나. 실험 결과

성능에 대한 시뮬레이션 결과는 [Fig. 16]와 같다. 갱신 즉시 저장 방식이 캐싱 후 저장하는 방식에 비해 성능이 크게 떨어지는 것을 알 수 있다. 또한 “총 처리시간 = DB 질의 평균 수행 속도 x (요청횟수 - 캐싱 횟수)” 로 동작함을 알 수 있다.



[Fig. 16] Chungmukong's Battle on the Sea

안정성에 대한 시뮬레이션은 서버 예외상황 시에 손실되는 데이터량에 따라 판단할 수 있다. 손실되는 데이터량은 곧 저장되지 않고 서버 메모리에만 누적되어 있는 데이터들의 수를 말한다. 시뮬레이션 결과는 [Fig. 17]와 같다. 캐싱 후 저장하는 방식은 캐싱하는 데이터 종류에 따라 누적량이 커지고 손실량 또한 증가함을 알 수 있다.



[Fig. 17] Chungmukong's Battle on the Sea

4. 결 론

본 연구는 온라인 게임에서 사용되는 서버의 데이터 저장 방식에 대해 제안하고 평가하였다. 이를 위해 데이터연동 계층을 정의하고 여러 저장 방식을 제안하였으며 분류하였다.

데이터연동 계층은 서버의 3계층 중 하나로 서버의 여러 데이터들을 데이터베이스에 저장하고 불러오는 역할을 하는 계층이다. 데이터연동 방식은 데이터 적용 방식에 따른 분류와 데이터 저장 시기에 따른 분류, 콘텐츠-원자성 보장 여부에 따른 분류, 입력 방식에 따른 분류로 4가지 기준을 가지고 나눌 수 있다. 데이터를 서버에 적용하는 방식에 따라 데이터 선적용 방식과 후적용 방식으로 나눌 수 있으며 데이터 저장 시기에 따라 즉시 저장 방식과 갱신 후 저장 방식으로 나눌 수 있다. 또 콘텐츠-원자성 보장 여부에 따라 보장 방식과 비보장 방식으로 나눌 수 있으며 데이터 입력 방식에 따라 절대값 입력 방식과 상대값 입력 방식으로 나눌 수 있다.

사례인 “충무공 해상대전”에 데이터 선적용 방식과 데이터 후적용 방식를 각각 적용하여 성능과 안정성을 비교하였다. 즉시 저장 방식과 갱신 후 저장 방식도 각각 적용하여 성능과 안정성을 비교하였다.

본 연구를 통해 온라인 게임의 서버 개발 시에 게임의 특징에 맞는 더 효과적인 데이터연동 방식을 선택할 수 있을 것으로 기대된다.

REFERENCES

[1] Deok-Ju Lee, Impact Assessment of online game regulation policy : “Shut-down” and “Tax”, ie Magazine Vol. 18, No. 3, pp.38-43, 2011. 09.

[2] 2013 White Paper on Korean Games, Guide to Korean Games Industry and Culture, Korea Content Agency, 2013. 10.

[3] Kengo Nakajima, Technology to support

online games, wikibook, 2012. 06.
[4] Seung-Woo Yun, Adoring-Lecture TCP/IP Socket Programming, Freelec, pp.505-517, 2003. 04.
[5] Jinseong Choi, Xuefeng Piao, Jaewoo Jeon, Samkweon Oh, A Comparative Analysis of Socket I/O models for Massively Multi-player On-line Network Game Server, Autumn Conference of Korea Information Processing Society, Vol. 9, No, 2, pp.1209-1212, 2002. 11.
[6] http://javawork.egloos.com/viewer/2210019, Korea Online GameServer Developer Group, 2009. 01.
[7] J.Y. Lim, I.K. Park, J.Y. Chung, K.H. Shim, "Technical Trend of Distributed Game Server", Electronics and Telecommunications Trends, ETRI, Vol. 20, No. 4, 2005. 08.
[8] Wen tong Cai, Percival Xavier, Stephen J. Turner, Bu-Sung Lee, "A Scalable Architecture for Supporting Interactive Games on the Internet", 16th Workshop on Parallel and Distributed Simulation, pp.60, 2002. 05.
[9] Müller, J., & Gorlatch, S. A Scalable Architecture for Multiplayer Computer Games, In GI Jahrestagung, pp. 154-158, 2004.
[10] Seungmo Koo, How to Support an Action-Heavy MMORPG from the Angle of Server Architectur, CGDC 2010, 2010. 06.
[11] Jeong-Hoon Kim, Efficient and scalable Online Game Server Architecture for Massive Users, Journal of Korea Game Society, Vol. 2, No. 2, 2002. 11.
[12] Jong-Gwan Choi, Hye-Young Kim, Won-Sik Woo, A Study of a Game User Oriented Load Balancing Scheme on MMORPG, Journal of Korea Game Society, Vol. 12, No. 3, 2012. 06.
[13] HyoJoo Pack, TaeYong Kim, Traffic Analysis and Modeling for Network Games, Journal of Korea Multimedia Society, Vol. 9, No. 5, pp. 635-648, 2006. 05
[14] Mark Claypool, Network Analysis of Counter-strike and Starcraft, In Proceedings of the 22nd IEEE International Performance,Computing, and Communications Conference (IPCCC), 2003. 04.
[15] Jong-Soo Kim, Tai-Suk Kim, A Study on a Database Design for the Multi-User Online Game, Spring Conference of Korea Information Processing Society, Vol. 12, No 1, pp.361-364, 2005. 05.
[16] Sam Kweon Oh, Min Seok Kang, "Performance Evaluation of Synchronization Algorithms for Multi-play Real-time Strategy Simulation Games", Hoseo The Journal of Research Institute for Engineering & Technology, Vol. 32, No 1, 2013. 06.
[17] Min Seok Kang and Kyung Sik Kim, "An Effective Scene Manager for the Online Game 'Chungmukong's Battle on the Sea'", 2009 Fifth International Joint Conf. on INC, IMS and IDC(NCM09), pp. 1267-1270, 2009. 08.
[18] Jung-Jun Lee, A Successful Case of the 'Great Admiral Yisunsin Festival' that Integrates 'Chungmukong E-sports Competition', The Journal of the Korea Contents Association, Vol. 11, No 6, pp127-134, 2011. 06.

강 민 석(Kang, Min Seok)

2005년 호서대학교 게임공학과 (학사)
2008년 호서첨단정보기술대학원 게임애니메이션과(석사)
2015년 호서대학교 컴퓨터공학과 (박사)
2009년-현재 KOG

관심분야 : 온라인 게임, 서버/DB 프로그래밍

김 경 식(Kim, Kyung Sik)

1982년 서울대학교 전산기공학과 (학사)
1984년 서울대학교 전산기공학과 (석사)
1990년 서울대학교 컴퓨터공학과 (박사)
1984년-1991년 한국전자통신연구원 선임연구원
1991년-현재 호서대학교 게임학과 교수

관심분야 : 기능성 게임, 게임 교육, 서버/DB 플밍