온라인 게임 시스템 서버 아키텍처 연구

최 성, 박헌용*
*남서울대학교 컴퓨터학과
e-mail:wmmmw@paran.com

Study on The Online Game Server Architecture

Choi Sung, Heon-yong Park*
NamSeoul University

요 약

온라인 게임 서버는 비용과 확장성의 이유로 분산 서버 구조를 택하고 있으며, 게임의목적에 따라 다양한 관점에서 서버 아키텍처를 설계한다. 대용량의 서비스를 하는 대규모 분산 시스템이자 네트워크 가상환경 예이다. 온라인 게임을 위한 서버 구조는 기술적인 면 만 아니라 기획적인 면에서도 영향을 받는다. 어떠한 온라인 게임을 만들 것인지에 따라 서버 구조는 달리 결정되며, 게임을 어떠한 모습으로 발전시킬 것인지에 따라 서버 구조 또한 변경되어야 한다. 본 논문에서는 온라인 게임 서버 구조를 설계하는데 있어 필요한 사항들을 기술하고, 이를 수용하는 서버 구조를 설계 연구하였다.

1. 온라인 게임 아키텍처 개요

1.1 아키텍처의 실행

일반적으로 이러한 아키텍처 작업은 모든 일의 시작인 "전략기획(Strategy Planning)" 과정에서 부터 진행되어야 한다. 아키텍처 작업의 결과 물은, "누가", "무엇을", "왜", "어떻게", "언제까지", "어느 곳에서" 해야 하는지를 계획 세우는 전략기획 과정의 밑그림이 된다. 하지만 여러 가지 여건상, 경영진과 전형적인 기획자, 마케팅 담당자 만이 참여하는 전략기획 팀이 많으며, 이러한 경우 "어떻게" 해야 하는지를 놓치기 쉽다. 아키텍처 작업은 전략기획 과정을 시작으로, 온라인 게임의 개발 전 과정에 걸쳐 지속적으로 진행된다

2. 네트워크 가상환경 설정

2.1 네트워크 가상환경 개요

네트워크 가상환경은 많은 사용자들이 실시간으로 연결되어 상호작용을 하는 소프트웨어 시스템이다. 일반적으로 사용자는 자신의 컴퓨터나 콘솔을 이용하여, 가상 환경의 내용에 대한 사용자 인터페이스를 제공받는다. 이러한 환경들은 사용자들이 충분히 몰입하여 체험할 수 있도록 사용자에게 실제적인 감각을 제공하려 한다. 실감나는 3차원 그래픽과 스테레오 사운드가 어우러져 제공되는 것이 하나의 방법이다. 네트워크 가상환경은 다음과 같은 다섯 가지 일반적인 특징으로 구별된다.

1) 공간에 대한 인식의 공유 모든 참여자들은 같은 공간에 존재한다는 공통된 환상을 가져야 한다.
2) 존재에 대한 인식의 공유 : 공유 공간에 들어갈 때, 각 참여자들은 아바타(avatar)라 불리는 가상의 인간을 소유. 각 참여자들은 서로의 아바타를 볼 수 있다.
3) 시간에 대한 인식의 공유 : 참여자들은 다른 이들의 행동이 일어난 그 시점에 이를 볼 수 있다.
4) 통신할 수 있는 방법 : 효율적인 네트워크 가상환경의 기본은 시각화 된 형태로 구성된다.
5) 공유할 수 있는 방법 : 위에 설명한 요소들은 높은 수준의 화상회의 같은 시스템을 위해 효과적으로 사용될 수 있다. 이 네트워크 가상환경은 많은 사용자들에게 그래픽을 통해 서로 간의 상호작용과 정보공유, 가상환경 속에 있는 물건들을 조작할 수 있도록 지원하는 시스템이다.

2.2 네트워크 가상환경의 설계 방안

1) 네트워크 대역폭으로 인한 공정성 문제 : 사용자들은 이질적인 접속 경로를 통해 Net-VE에 접속하며, 이로 인한 속도의 차이를 감출 것인지, 인정하고 받아들일 것인지 결정해야 한다.
2) 공정성을 고려한다면, 네트워크 성능의 요구 수준이 실제 사용자들이 연결된 네트워크 성능을 넘지 않는 범위 내에서, 최소한의 공통 분모를 찾아내야 한다. 이를 찾아내어 시스템의 이질성을 감추도록 노력한다면, 잘 설계된 Net-VE라 할 수 있다.
3) 다른 대안으로 사용 가능한 모든 자원을 모두 이용하는 것이다. 하지만 이 방법을 선택한다면, 설계자는 공정한 진행의 문제에 직면한다. 가상 환경에 대등한 입장으로 참여하고 있는 사용자들에게, 사용자들이 사용하는 네트워크나 장비의 성능에 따라 다른 수준의 정보들이 전달되기 때문이다.
4) 장비의 이질성으로 인한 공정성문제 : 실제 운영되는 Net-VE를 보면, 사용자들이 똑같은 장비를 사용하지는 않는다. 장비의 이질성은 그래픽 디스플레이나 계산, 오디오 성능 등에 영향을 미친다.
5) 단일 이미지를 위한 분산된 상호작용 : 분산 상호작용은 Net-VE의 질을 결정한다. Net-VE 시스템은 가상 환경 전체가 사용자의 로컬 머신에 있다는 환상을 사용자에게 제공, 사용자의 행동들이 즉각 가상 환경에 영향을 미친다고 느껴야 한다.
6) 네트워크 지연의 고려
   가) 시스템 내부에서 정보를 교환하기 위한 메시지가 단일 시스템이라는 환상을 유지하는 것은 힘든 일이다. 메시지가 전송되어 실제로 도착하는데 걸리는 지연 시간을 네트워크의 물리적 특성상 피할 수 없다. 전송 시간은 메시지를 송수신하는 호스트의 위치와 네트워크의 종류에 따라 다르게 걸린다. 모든 호스트는 Net-VE의 현재 상태를 일관되게 실시간으로 보여줄 수 있도록 해야한다. 또한 이러한 네트워크 지연으로 호스트들이 받는 가상 환경에 대한 정보들은 항상 시간이 지난 것일 수 밖에 없다. Net-VE의 설계자는 반드시 이를 고려하여야 한다.
   나) 특히 여러 명의 사용자나 컴포넌트들이 직접적으로 상호작용을 할 때, 이러한 네트워크 지연을 다루기가 더욱 힘들다. 예를 들어, Net-VE 시스템은 가상환경의 참여자끼리 발생할 수 있는 충돌에 대한 체크와 이에 대한 동의(agreement), 해결을 정확히 처리해야 한다. 정확한 충돌 체크어떠한 시점에서도 다른 사용자에 대한 정확한 위치 정보를 가지고 있는 사용자는 단 한명도 없기 때문에 더욱 어렵다. 네트워크 지연은 전송 받은 모든 정보는 모두 시일이 지났다는(out-of-date) 사실이다. 그러므로 메시지가 전송되는 사이에 충돌을 피해 이동하였는데도, 네트워크 지연에 의해 시간이 지난 오래된 정보에 근거 충돌이 일어났다고 생각하는 호스트도 있다. 또 다른 사용자는 그 메시지의 전송 시간이 다르게 되어서, 다른 판단을 할 수도 있다. 충돌이 여부에 대해 확신이 있다 하더라도, 그 충돌의 정확한 발생시점이 판단되어야 한다. 또한 이러한 물리적으로 충돌된 참여자들에게 어떠한 영향을 미치게 되는지도 결정되어야 한다. 가상 환경 내에서 이러한 충돌이 두 개 이상의 객체와 관련고 관련된 객체의 수가 많아질 수록, 판단하고 결정해야 할 사항들은 비약적으로 복잡해진다.
   다) 이러한 직접적인 상호작용은 Net-VE 시스템에서 매우 빈번하게 일어난다. 게임에서는 이러한 충돌이, 총알을 과녁에 맞추었는지 판단하는 기준이 된다. 공학적인 시뮬레이션에서는 차나 비행기 등이 도로나 바람과 어떻게 상호작용을 하는지, 충돌을 고려하여 정확히 계산해 낸다. 이러한 상호작용에는 마찰력과 열, 심지어 청각적인 정보까지 포함된다. Net-VE 시스템의 분산 상호작용은 오디오 커뮤니케이션의 전송으로 더욱 복잡해진다. 가상 환경에서 발생되는 청각 정보를 들어야 하고 소리에 근거하여 다른 사용자나 물체들의 거리를 인지하도록 하기 때문이다.
7) 실시간 시스템 설계 및 자원의 관리 : 실시간 상호작용은 Net-VE 응용 프로그램의 프로세스 및 쓰레드 아키텍처를 정의하는데 있어 중요한 고려사항이다. 많은 태스크들이 동시에 CPU를 할당, 즉 실행되기 위해 경쟁한다. 일반적인 시스템들과는 달리, Net-VE의 모든 태스크들은 높은 수준의 실시간 제약을 가진다. Net-VE 설계자는 다양하고 많은 태스크들의 실시간 요구를 만족시

키도록 해야한다.

가) 먼저, Net-VE는 로컬 사용자의 실시간 상호작용을 지원해야 한다. 사용자들이 키보드와 마우스, HMD 등을 이용하여 행하는 행동들은 재빨리 감지되고 처리되도록 배려되어야 한다. 예를 들어, 그래픽 이미지가 초당 30 프레임의 일정한 속도로 생성되어야 한다고 했을때, 이미지의 생성이 지연되면 화면은 끊어지듯 보이게 되며 Net-VE에 몰입하는 체험감은 떨어진다. 변동이 없다면, 모든 사용 가능한 CPU 사이클은 높은 수준의 이미지를 생성하는데 쓰여지는게 보통이다. 또한 네트워크 패킷은 비동기적으로 도착하며 이들은 가능한 빨리 처리되는 것이 좋다. 이유는, 이들의 처리가 늦어질수록 전송 지연이 커지는 것이 되며, 다른 참여자들에 대해 덜 정확한 정보를 얻게 되는 것이기 때문이다. 마지막으로, 물리학적인 모델링과 충돌 체크 또한 초당 몇번 정도는 수행되어야 한다.

8) 확장성 : Net-VE의 Scalability(확장성) 또는 크기(size)는, 시스템에 동시에 참여할 수 있는 개체(entity)의 수로 측정된다(구체적인 예로 시스템이 수용 가능한 사용자의 수, 시스템이 처리 가능한 처리단위의 수등을 기준으로 한다. 이경우 scalability가 높다라는 것은, 사용자나 처리단위들을 더 많이 수용하도록 요구되는 상황에서, 시스템이 이를 쉽게 수용할 수 있다는 의미이다.)

가) Net-VE의 개체는 Net-VE에 참여하는 호스트에 의해 각각 모델링 되는 객체(object)들이다. Net-VE의 개체들은 사람에 의해 조종되거나 컴퓨터에 의해 조종되는 탈것(vehicles), 나무, 바위 등을 포함하는 지형, 현재 날씨와 객체 그룹과 같이 논리적인 객체들까지도 포함할 수 있다. 단순한 게임시스템에서는 단 두 개의 개체만이 필요할 수도 있으며, 복잡한 공학 시스템이나 시뮬레이션 시스템에서는 수백만개의 개체가 요구되기도 한다. 이처럼 시스템이 요구하는 scalability의 범위는 다양하다. Scalability라는 개념은 상황과 목적에 따라 다르게 측정될 수 있다. Net-VE에 동시에 접속한 호스트의 수나, Net-VE 시스템에 접속한 사용자들의 물리적인 거리라 볼 수도 있다.

나) Net-VE의 scalability는 다양한 요소에 의존적이다. 예를 들면, 네트워크의 용량, 프로세서의 성능, 렌더링 속도, 공유 서버의 성능과 효율들이 중요한 기준이 된다. Scalability를 얻는다는 것은 비용이 많이든다. 왜냐하면 scalability는 실질적으로 Net-VE 시스템의 모든 측면을 향상시켜야 가능하기 때문이다.

3. 온라인 게임설계 범위

3.1 온라인 게임의 서버 아키텍처

1) 중앙 집중 방식 : 설계와 개발에 용이하나, 고비용과 확장성의 단점이 있다.
2) 명시적인 지역 분할 방식 : 가상 공간을 지역적으로 분할하여, 각 서버가 관리권을 맡는 방식이다.
3) 묵시적인 지역 분할 방식 : 서버가 가상 공간을 분할하여 관리하기는 하지만, 관리영역에 따라 가상 공간을 기획적으로 분할하지는 않은 경우이다.
4) 분할 경계에 대한 복제 기법 : 서버가 관리해야 하는 영역들의 경계 셀들을, 복제하여 관리해야 하는 영역으로 지정하고, 이에 대한 관리 권한을 특정 서버에서 두어, C/S 방식으로 동기화 하는 것이다. 이를 잘못 구현하는 경우, 아이템 복제 등의 부작용이 생길 수 있다.
5) 지역 분할 동기화 방식 : 가상공간을 지역적으로 분할하여, 각 서버가 관리권을 맡는 방식이다.
6) 부하 분산 방식 1 : DB나 가상공간 관리자를 통해 접근을 하는, 중앙 집중식 상호 배제(ME). 각 서버는 클라이언트의 Agent 역할에 집중하며, 가상공간 운영을 위한 서버가 따로 존재해야 한다. 서버에만 DB가 존재하며, 이에 대한 상호 배제를 위해서 Semaphore를 제공하게 된다.
7) 부하 분산 방식 2 : 각 서버의 통신을 통해 접근을 제어하는 분산 방식 상호 배제(ME). 각 서버가 가상공간을 같은 관점에서 운영, 상호간의 통신을 통해 동기화를 이룬다. 동기화 방식에 따라 P2P 갱신 방법과 C/S 갱신 방법으로 나뉜다.
8) 부하 분산 방식 3 : 부하분산방식 2와 동일하나, 서버가 지역적으로 서버가 분산되어 있는 네트워크 가상환경의 경우이다. 서버들은 Internet을 통해 연결. 동기화 방식에 따라 P2P 갱신 방법과 C/S 갱신 방법으로 나뉜다.

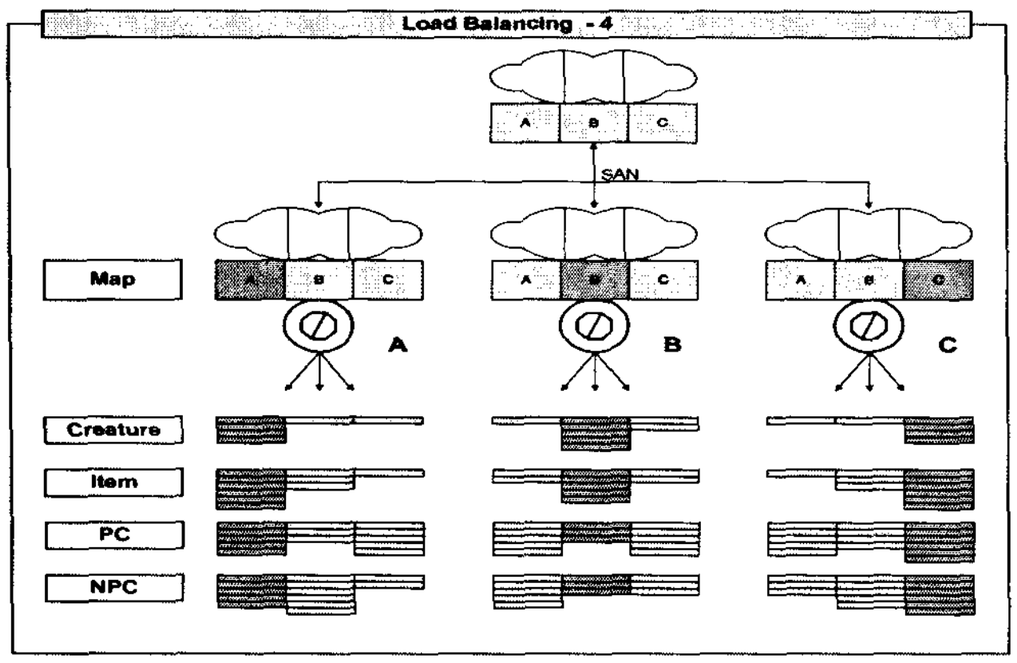

그림 1. 부하분산방식의 구성

3.2 네트워크 가상환경의 서버 아키텍처

1) 중앙 집중 방식 (Centralized)

2) 지역 분할 방식 (Region Partition)

가) NetEffect : 지역 분할 방식이지만 서버들이 인터넷상에 분산되어 있기 때문에, 구조상 단점이 있다.

3) 지역 분할 동기화 방식

가) DIVE : 신뢰성 있는 멀티캐스트 프로토콜을 사용한 P2P 갱신 방법을 사용한다. 동적 DB를 사용하여, 가상 환경의 변화에 유연하다.

나) BrickNet : 데이터를 분할하여 각 서버에 분산시키는 방법이며, DB의 관리는 중앙 집중식으로 (C/S 갱신 방법) 이루어진다.

4) 부하 분산 방식

- VISTEL : 중앙집중식 서버에서 DB를 관리하고, 변경사항에 대한 저장과 변경사항의 재분배를 서버에서 전부 맡는다.

5) 부하 분산 방식

- SIMNET : P2P 갱신 방법 사용한다. 정적 DB를 사용하여 가상환경의 내용이 변경되는 경우, 모든 서버의 DB 구조를 재구성해야 하는 단점이 있다.

4. 온라인 게임서버 아키텍처 설계

몇가지 서버 구조의 예를 통하여 지적한 사항들을 점검하고 이를 바탕으로 새로운 구조를 만든다.

4.1 지역분할 구조를 택한 온라인 게임의 서버 구조

각 서버 간 운영권이 독립적이며, 동기화 비용이 적은 구조이다. 이 구조에서 프로세싱 서버의 성능과 효율성을 극대화할 수 있는 방안을 제안한다. 앞에 설명한 온라인 게임의 엔진구조를 기반으로 생각해 볼 때, 프로세싱 서버가 감당해야 하는 역할은 크게 네 가지 이며, 이는 FT, AT, PT, NT로 대표된다. 우리는 이 네 가지 역할을 순차적으로 처리되도록 구성할 수도 있고, 완전히 병렬적으로 실행되도록 구성할 수도 있다. 여기서는, 각 쓰레드들을 Work-Crew 쓰레드 모델처럼 순차적으로 구성하되, 파이프라이닝(pipelining) 가능한 구조로 배치하였다. 이러한 구조에서 다음과 같은 개선안을 제안한다.

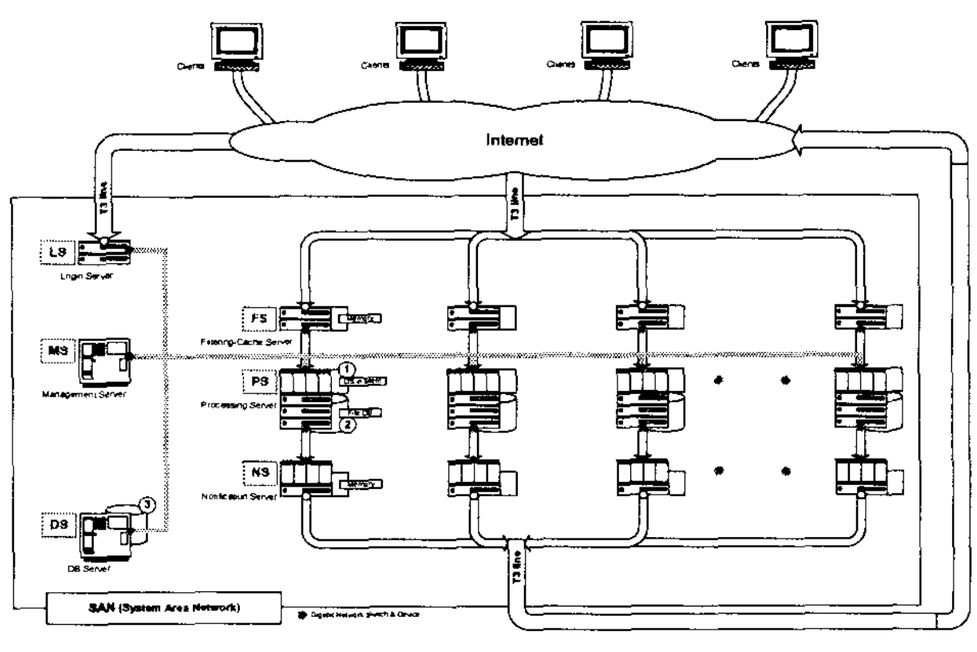

그림 2. 지역분할 구조의 게임서버 구조

1) SAN(Server Area Network)의 구조

가) PS가 해야하는 네 가지 역할 (FT-AT-PT-NT) 중, FT와 NT의 역할을 서버로 분리해 내었다. FS, PS, NS가 하나의 관리 영역을 운영하는 서비스 스트림(Service Stream)이 된다.

나) Gigabit Network을 SAN의 Backbone으로 도입한다.

다) FS와 PS, PS와 NS는 네트워크를 직접 연결한다. 또는, FS/PS/NS를 Loosely-Coupled Multiprocessor 에 올리는 것도 좋은 방법이다.

라) AS : AT의 역할을 AS라는 또 다른 서버로 분리시켜 낼 수도 있으나, 기획에 따라 동기화 비용에 대한 검증이 필요하다.

2) Filtering Server (Cache Server)

가) FS는 PS가 담당하는 Client들과 연결을 유지하고 Packet을 전송 받아, PS가 원하는 형태로 PDU를 처리하여 넘겨주는 역할을 한다.

나) 목적 1 : 네트워크 프로토콜에서 발생하는 오버헤드(Protocol Overhead)를 감당해주는 역할. TCP의 경우 통신 중에 발생하는 문제들을 해결하기 위해 부가적인 처리들이 많다. 이 역할을 위해서는 FS 에 많은 메모리가 있어야 하며, 고속의 프로세서가 필요하다.

다) 목적 2 : 네트워크 입력의 처리 효율 증대. 2개의 Input Process가 동시에 프로토콜 오버헤드를 감당한다. 이를 위해서는 앞 단에 2개의 Network Device개 있어야 하며, 시스템은 2-Way Processor여야 하고, 네트워크 라인 역시, Dedicated Line이어야 한다.

3) Notification Server

가) 서버-> 클라이언트 간 빠른 Snapshot 정보 전송을 위한 Multicasting 방법을 서버 구조에서 방법을 수용한다.

나) PS는 클라이언트로 보내야 하는 주기적인 정보들을 큰 덩어리로 NS로 전송하며, NS는 이를 받아서 재조합하거나 각 클라이언트에 맞게 생성하여, 일일이 클라이언트에게 전송한다.

다) PS에서 부담해야 하는 PDU 생성시간, 전송시간을 NS가 부담하기 때문에 PS의 부담이 덜어지며, 또한 NS에서는 PDU를 압축할 시간도 벌 수 있다.

라) 참고사항 : 가상 공간내의 객체 관리 방식

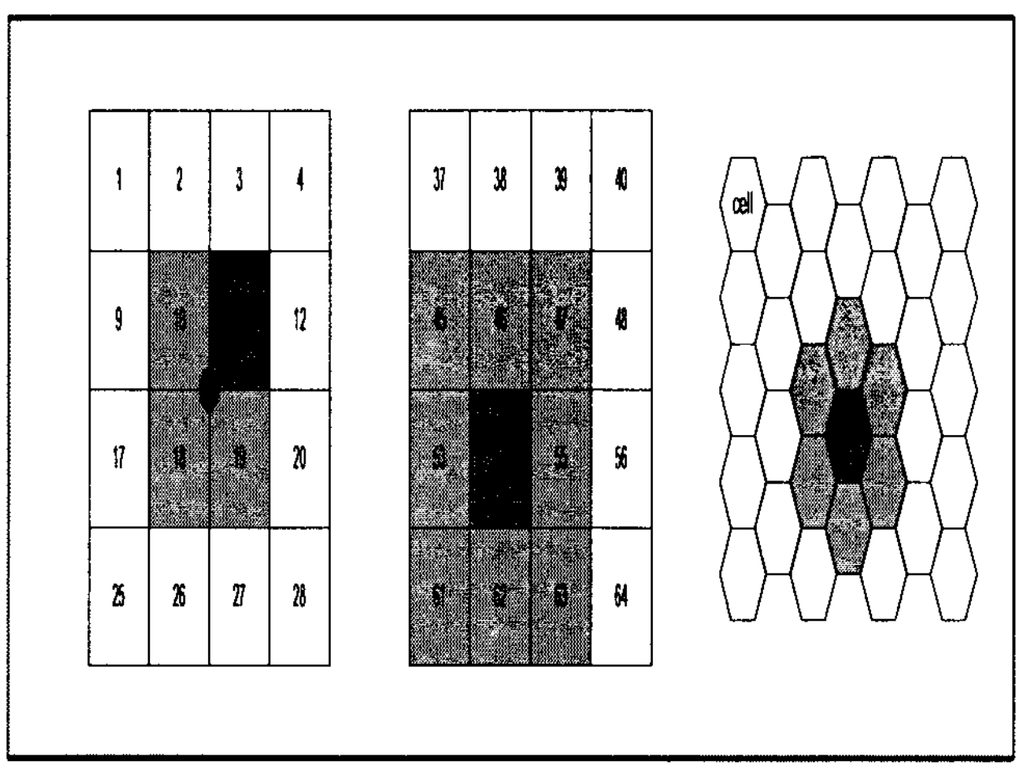

그림 3. 가상 공간의 동기화 공간 분할 방식

(1) Map Grid에 기반하여 객체의 위치정보 유지 (공간기반 배열방식) : 원하는 영역만큼 빠르게 선택가능.

(2) 객체의 좌표를 기준으로 2쌍의 리스트 유지 (객체기반 정렬방식) : 각 객체는 xprev, xnext, yprev, ynext 값을 가지며, 위치를 기준으로 리스트로 연결되어 있다. (3차원일 경우 z축은 리스트로 관리하거나 속성 값으로 관리 할 수 있다.) 원하는 영역만큼 빠르게 찾아 선택이 가능하지만, 분산 환경에서는 적합하지 않다. (공통되는 GDS를 구성하는 데는 적합하지 않다.)

마) 참고사항 : 가상공간의 동기화 공간 분할 방식

(1) 4 Cell : PC로부터 인접한 Red-Point를 중심으로 4개의 Cell 선택

(2) 9 Cell : PC가 위치한 Cell을 중심으로 9개의 Cell 선택

(3) 7 Cell : PC가 위치한 Cell을 중심으로 7개의 Cell 선택 (NPSNET)

4.2 온라인 서버 구조

다음은 다양한 서비스와 연계되는 온라인 게임의 서버구조를 예로 든 것이다. 온라인 게임 뿐만아니라, 웹(기본 서비스), 웹 기반의 커뮤니티, 광고, 메신저, 과금 등의 고객 서비스가 필요한 경우이다. 또 게임이 마케팅적인 요구를 수용하는 경품 게임의 경우에는, 마케팅 분석을 위한 통계자료와 이에 대한 관리 도구가 중요한 요구 사항이 된다. 시스템 관리와 서비스 운영을 위한 도구와 이를 위한 협업 요구사항도 간과해서는 안 된다.

이와 같이 다양한 서비스들이 연계되는 경우, 시스템의 성능과 효율보다는 구조적인 관점에서 체계를 잘 잡는 것이 중요하다. 다양한 프로세스 시나리오와 데이터 연계가 요구되기에 시스템의 복잡성이 높아지며, 구조를 잘 잡아야만 시스템의 안정성을 보장할 수 있다. 또한 새로운 개발 요구에 대한 확장성 역시 보장하기 더욱 어려워지므로, 구조적인 관점은 매우 중요하다.

이와 같이 다양한 구성 요소를 연계하는 방법을, 응용시스템 통합 전략 (Application Integration Strategy : AIS)이라 부른다. 기획팀, 마케팅 팀, 서비스 운영 팀 등 시스템을 사용하는 사용자들과, 실제 사용 시나리오를 고려하여 시스템을 설계하는 것이 AIS의 기본 작업이다. 기획 및 설계과정에서, 특히 데이터 모델링과 이를 협업하여 처리하는 프로세스가 명확하게 도출되어야 하는 것이다. 기술적인 측면에서는 각 구성 요소들의 연계 수준은 다음과 같다.

1). Synchronization IPC를 통한 연계

2). Communication IPC를 통한 연계

3). DB를 통한 실시간 연계

4). DB를 통한 비실시간 연계 : 통계/로그 데이터를 주기적으로 이용하는 경우 (1/day)

어떠한 시스템이던 데이터 모델링 과정이 제일 중요하다. 하지만 이러한 시스템의 경우에는 더욱 중

요하다. 위의 예처럼 다양한 구성 요소를 가지는 경우, 개발 관점에서 볼 때 일반적인 IT 시스템과 다를 바 없다. IT 시스템은 명확하고 간결한 구조 하에서 빠르게 개발하고, 쉽게 재설정하고 운영하는 것이 중요하다. 대신 온라인 게임들에서 요구되는 성능 측면은, 비용으로 보완된다. 때문에 온라인 게임의 서버 아키텍처를 설계하는 데 있어, 다양한 서비스와의 연계가 요구된다면, 효율성만큼이나 구조적 안정성을 고려하여야 한다. 그 구조적 안정성 이라는 것은, 명확한 협업 시나리오 설계연구에서 개발되어 진다.

5. 결론

기존 온라인 게임 개발 과정에서 묵시적으로 존재하던, 아키텍처링(Architecturing) 작업을 명시화 하여 하나의 과정으로 인식할 필요가 있다.
온라인 게임의 서버 아키텍처의 목표는, 빠르고 안정적인 서비스가 운영되도록, 서버 게임엔진을 뒷받침 해준다. 온라인 게임의 서버 아키텍처는 설계, 기획, 마케팅, 경영 여건을 고려하여 결정되어야 한다. 또한 다른 아키텍처와 연계하여 같은 연장선상에서 설계되어야 한다. 다양한 서비스들과 온라인 게임이 연계되는 경우에는, 효율성과 함께 구조적 안정성을 고려하여 서버 구조를 설계하여야 한다.

참고 문헌

[1] 신동원. 온라인 게임 배틀 서버 설계서
[2] 김상균. 대규모 네트워크 가상환경을 위한 동적 로드 밸런싱
[3] Michael Zyda. Networked Virtual Environments
[4] 채의근. A Survey on Networked VR Systems
[5] 조재호 역. TCP tuning guide for distributed application on wide area network
[6] http://synczone.net
[7] http://vrrc.kaist.ac.kr