

게임서버프로그래밍

2019년 2학기

한국산업기술대학교
게임공학부

정내훈

4장 게임 서버와 클라이언트

- 1 | 패키지 게임에서 게임 서버
- 2 | 온라인 게임에서 게임 서버
- 3 | 서버의 역할
- 4 | 게임 클라이언트와 서버의 상호 작용
- 5 | 게임 서버가 하는 일
- 6 | 게임 서버의 품질
- 7 | 플레이어 정보의 저장
- 8 | 서버 구동 환경
- 9 | 서버 개발 지침
- 10 | 더 읽을 거리

4.1 | 패키지 게임에서 게임 서버

- 패키지 게임
 - 소규모의 플레이어가 같이 모여서 하는 온라인 게임. Offline플레이가 가능한 경우도 많음, MOG(Multiplayer Online Game)이라고도 함
- 서버

플레이어 세 명 이상이 게임을 하려면 그들의 게임 플레이 상태를 저장하는 곳이 필요하고, 이를 위해 플레이어 중 한 명의 컴퓨터가 모든 플레이어의 게임 플레이 상태를 취합해서 유지하는 역할을 해야함 그곳을 서버(server)라고 부름

패키지 게임에서 게임 서버가 하는 역할은 지금 플레이어가 두세 명 혹은 많아도 십여 명 참여하여 게임 플레이를 하는 상태, 즉 세션(session) 처리를 담당하는 것

데디케이트드 서버 (dedicated server) : 클라이언트 프로그램과 같은 엔진을 사용하지만 렌더링과 사용자 입력 처리를 받지 않고, 순전히 클라이언트의 연결을 받는 세션을 처리만 하는 프로그램이 따로 운용하는 경우

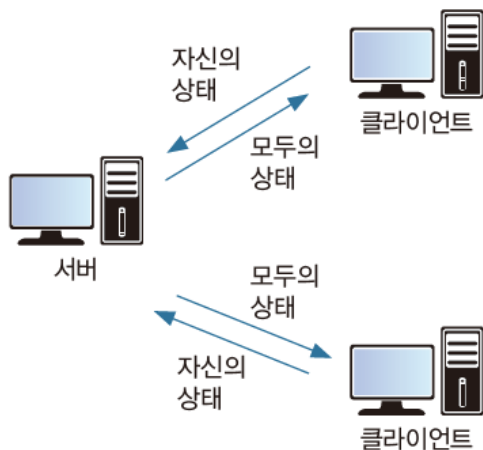


그림 4-2 패키지 게임에서 게임 서버와 클라이언트



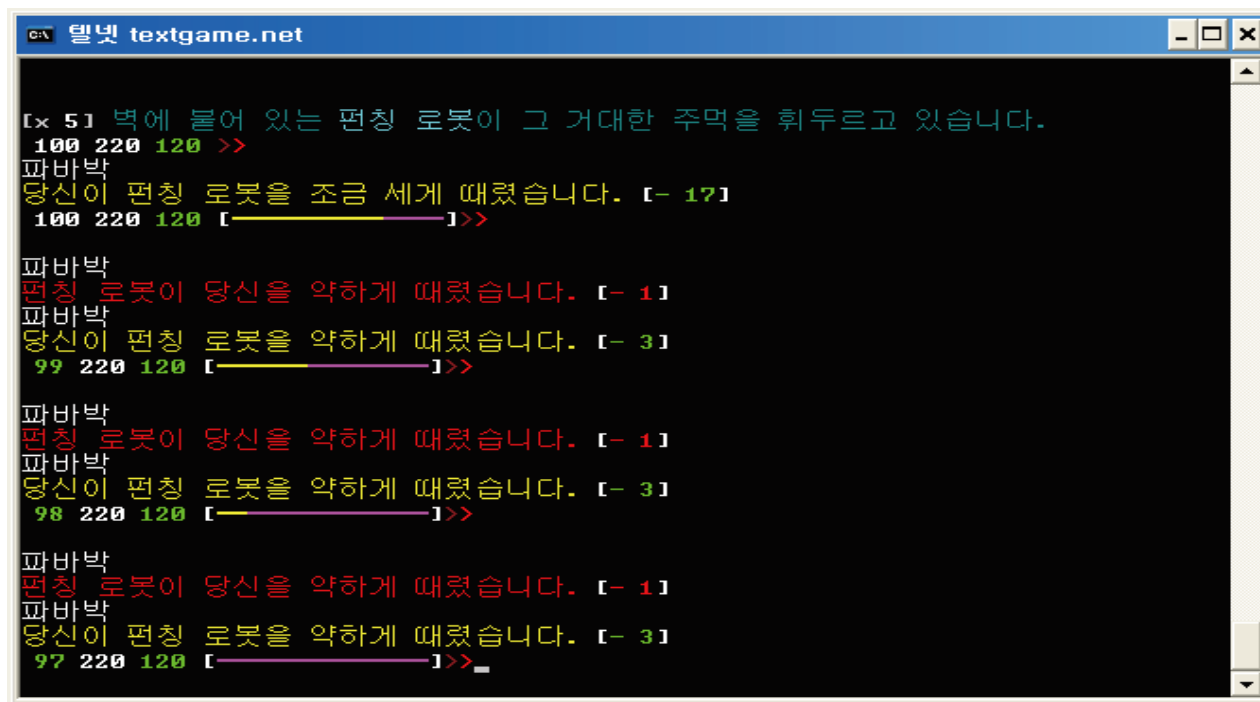
그림 4-3 Quake의 데디케이트드 서버

4.2 | 온라인 게임에서 게임 서버

- MUD(Multi User Dungeon)

여러 사람이 동시에 던전에 들어가서 모험하는 방식의 게임.

이때 MUD 게임은 그래픽 화면을 보여 주는 클라이언트 프로그램 없이, 텍스트 입출력만 받을 수 있는 콘솔(터미널)을 이용해서 게임을 즐기는 방식으로, 온라인 게임에서 본격적인 게임 서버는 이때부터 시작된다.



```
c:\> 텔넷 textgame.net

[ x 5 ] 벽에 붙어 있는 펀칭 로봇이 그 거대한 주먹을 휘두르고 있습니다.
100 220 120 >>
파바박
당신이 펀칭 로봇을 조금 세게 때렸습니다. [- 17]
100 220 120 [ _____ ]>>

파바박
펀칭 로봇이 당신을 약하게 때렸습니다. [- 1]
파바박
당신이 펀칭 로봇을 약하게 때렸습니다. [- 3]
99 220 120 [ _____ ]>>

파바박
펀칭 로봇이 당신을 약하게 때렸습니다. [- 1]
파바박
당신이 펀칭 로봇을 약하게 때렸습니다. [- 3]
98 220 120 [ _____ ]>>

파바박
펀칭 로봇이 당신을 약하게 때렸습니다. [- 1]
파바박
당신이 펀칭 로봇을 약하게 때렸습니다. [- 3]
97 220 120 [ _____ ]>>
```

그림 4-4 텍스트로만 표시되는 MUD 게임

4.3 | 서버의 역할

- 싱글플레이 게임을 처리하기(게임 루프)

입력받기: 키보드, 마우스, 터치 스크린, 마이크, 카메라 등으로 컴퓨터가 정보를 획득하는 과정.

게임 로직 처리하기: 게임 정보를 담고 있는 상태인 세션은 보통 1초에 60번 상태 변화를 한다. 상태 변화를 하는 과정을 게임 로직 처리라고 한다. 게임 로직 처리 과정 중에는 게임 플레이 판정, 가령 플레이어가 어느 캐릭터에 대미지를 주었는지 혹은 캐릭터가 어느 몬스터 캐릭터에 대미지를 받았는지 등을 계산한다.

렌더링: 변화된 상태를 화면에 표현한다.

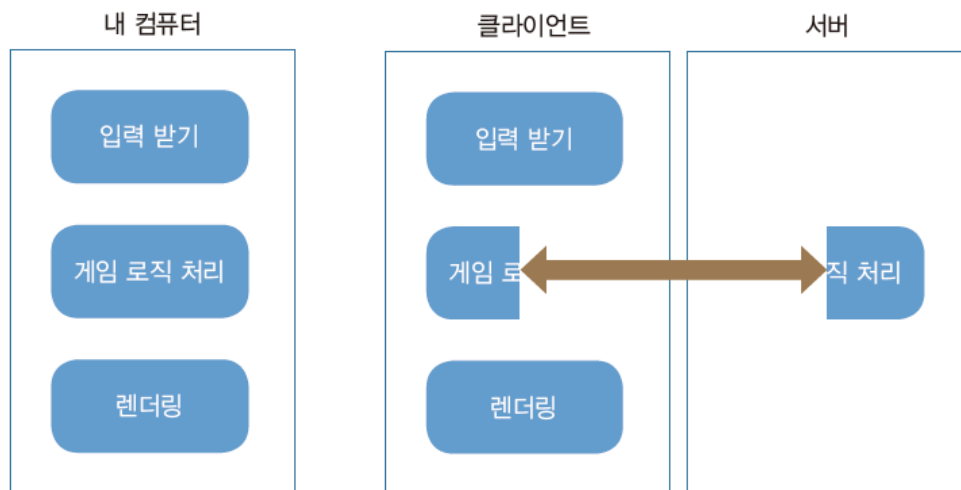


그림 4-5 게임 루프

클라이언트 : 1인용 게임에서 플레이어가 직접 만지는 컴퓨터.

대부분 온라인 게임에서는 클라이언트에서 게임 로직을 처리하는 역할 일부를 떼어 내 서버로 옮긴다. 그리고 클라이언트와 서버 간에는 컴퓨터 네트워크를 통해 서로 데이터를 주고받는다.

그림 4-6 게임 루프의 일부를 서버가 가져간 상태

4.4 | 게임 클라이언트와 서버의 상호 작용

- 게임 클라이언트와 서버의 상호 작용

연결	요청-응답	능동적 통보	연결 해제
<p>최초로 클라이언트가 서버와 데이터를 주고받을 준비를 하는 것</p> <p>서버는 클라이언트의신원을 확인하여 연결을 계속 유지할지 아니면 추방(연결 해제)할지 판단해야 한다</p>	<p>클라이언트는 서버에 메시지를 보내고, 서버는 이를 처리한 후 결과를 응답해 줌</p>	<p>클라이언트에서 요청을 보낸 적도 없는데 서버에서 능동적으로 통보</p>	-

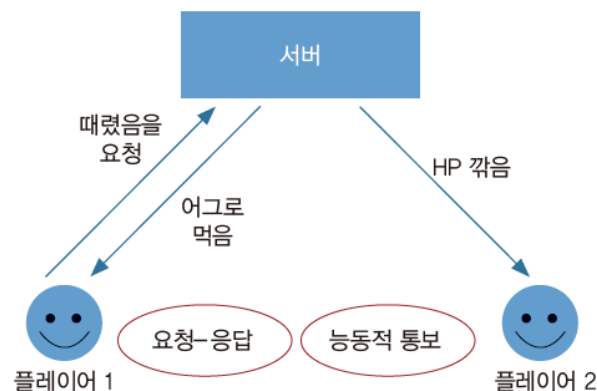


그림 4-7 두 클라이언트와 서버 간 상호 작용

4.5 | 게임 서버가 하는 일

- 게임 서버가 주로 하는 일

여러 사용자와 상호 작용

클라이언트에서 해킹 당하면 안 되는 처리

플레이어의 상태 보관

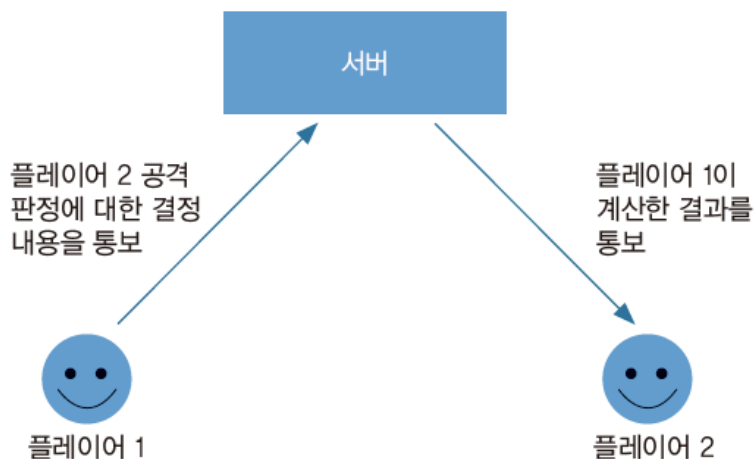


그림 4-8 클라이언트에서 일방적으로 판단하는 경우

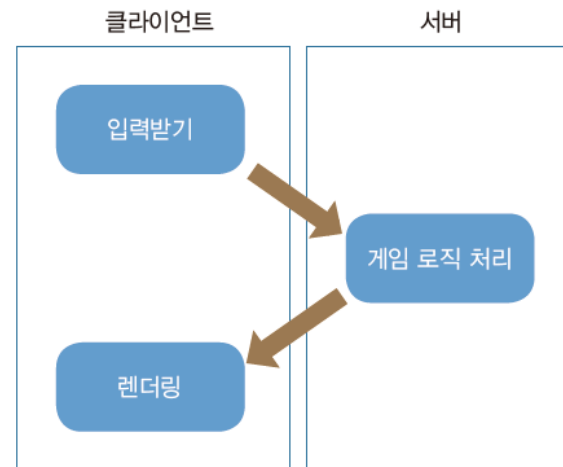


그림 4-9 게임 로직을 서버에서만 처리

4.6 | 게임 서버의 품질

4.6.1 안정성

“게임 서버가 얼마나 죽지 않는가”를 의미.

소프트웨어 측면에서 안정성에 악영향을 주는 주된 요인은 버그임. 구조적으로 설계가 잘못되었거나 사소한 코딩 실수가 서버의 안정성을 위협함.

안정성에는 “게임 서버가 얼마나 오작동을 하지 않는가”도 포함됨.

```
Mob mobs[100];
Player_AttackMob(int mobIndex)
{
    // mobIndex가 범위를 벗어난다면?
    mobs[mobIndex].m_hp -= 10;
}
```

개발 과정에서 안정성을 위해 노력할 것

1. 치밀한 개발과 유닛 테스트	2. 80:20 법칙	3. 1인 이상의 코드 리뷰	4. 가정하지 말고 검증하라
-------------------------	----------------	-----------------------	-----------------------

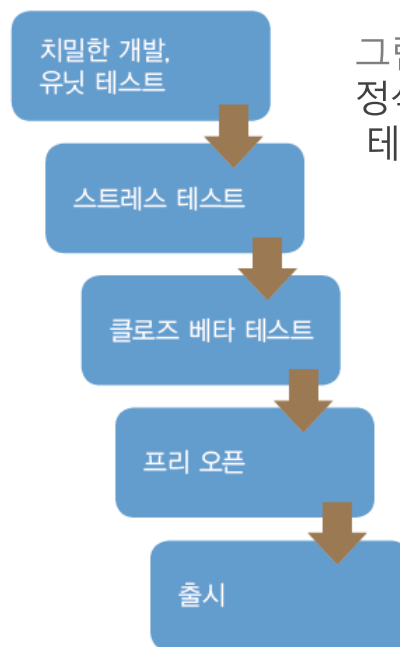


그림 4-10
정식 출시까지 거치는
테스트

4.6 | 게임 서버의 품질

- 서버의 불안정을 극복하는 방법
 1. 서버가 죽더라도 최대한 빨리 다시 살아나게 한다.
 2. 서버는 죽더라도 최대한 적은 서비스만 죽게 한다.
 3. 서버 오작동에 대해서 기록(로그)을 남길 수 있게 한다.

4.6 | 게임 서버의 품질

4.6.2 확장성

서버를 얼마나 많이 설치할 수 있느냐를 의미.

게임 사용자 측면에서 '사용자수가 늘어나더라도 서비스 품질이 떨어지지 않고 유지되느냐'가 곧 확장성을 의미한다.

서버 확장성을 올리는 방법에는 크게 수직적 확장(scale-up)과 수평적 확장(scale-out)이 있다.

구분	수직적 확장	수평적 확장
확장 종류	서버 머신의 부품을 업그레이드 혹은 서버 머신안의 CPU, RAM을 증설한다. 서버 프로그램을 최적화 한다.	서버 머신의 개수를 증설한다.
서버 소프트웨어 설계 비용	HW 확장: 없다. SW 최적화 : 매우 높다.	높다.
확장 설치 비용	높다(기하급수적으로 높아진다).	낮다(선형적으로 높아진다).
과부하 지점	서버 컴퓨터 자체	네트워크 장치
오류 가능성	HW : 낮다 SW : 매우 높다(고도화된 멀티 쓰레드 프로그래밍 필요).	높다(여러 머신에 걸쳐 비동기 프로그래밍 방식으로 작동하므로).
단위 처리 속도 요구량	높다(로컬 컴퓨터의 CPU와 RAM만 사용).	낮다(여러 서버 컴퓨터 간의 메시징이 오가면서 처리하므로).
확장 가능 총량	낮다(서버 컴퓨터 한 대의 성능만 사용하므로).	높다(여러 서버 컴퓨터로 부하가 분산되므로).

표 4-1 수직적 확장과 수평적 확장

4.6 | 게임 서버의 품질

• 4.6.3 성능

성능은 기본적으로 얼마나 빨리 처리하는지를 의미한다.

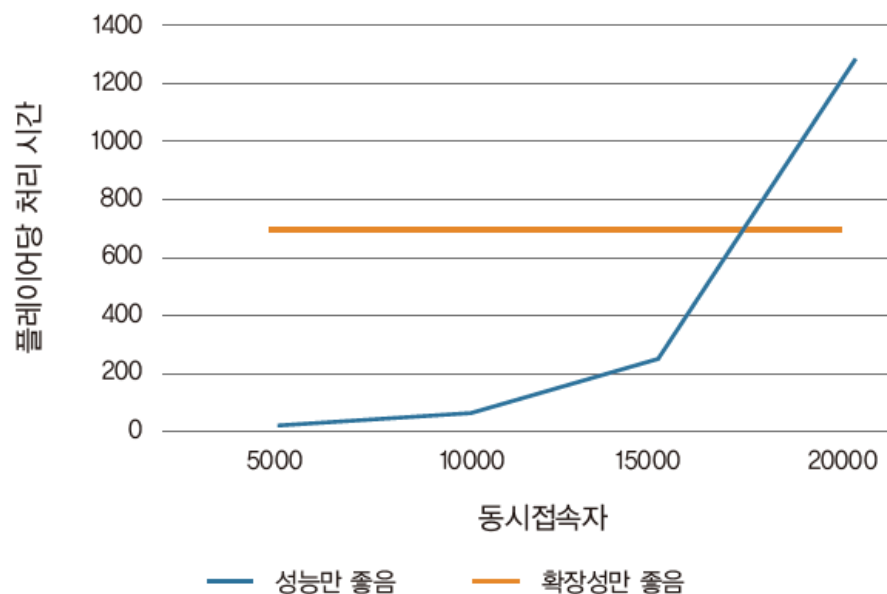
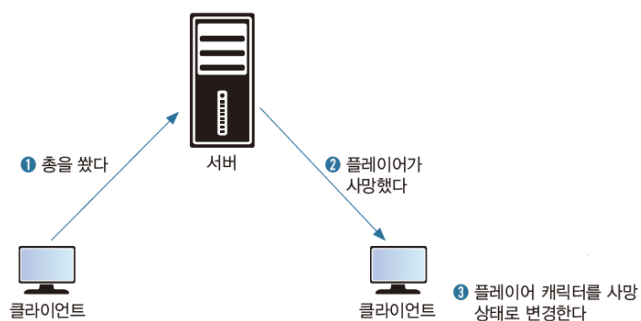


그림 4-12 성능만 좋거나 확장성만 좋을 때 그래프

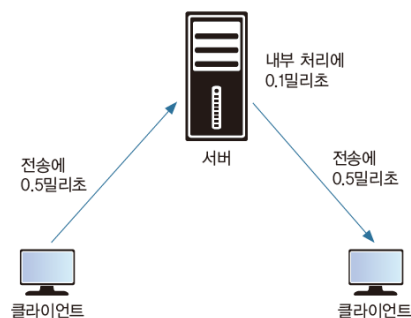
4.6 | 게임 서버의 품질

• FPS 게임에서 처리 과정



- 1 플레이어가 저격총을 쏘는 순간 클라이언트는 서버에 총알을 쏘았다는 의미의 메시지를 보낸다.
- 2 서버는 이를 받아서 메시지를 처리, 그리고 "헤드샷 때문에 플레이어가 사망했다."와 같은 메시지 처리 결과를 다른 플레이어에게 보내야 한다.
- 3 클라이언트에서는 이 메시지를 받고 플레이어가 죽는 모습으로 바뀌어야 한다.

그림 4-13 FPS 게임에서 처리 과정



총 걸리는 시간은 1.1밀리초

→서버가 메시지 1개를 처리하는 데 0.1밀리초가 걸린다면, 1초에 최대 처리할 수 있는 메시지 수가 1만 개임을 의미하므로 0.1밀리초는 게임 서버 입장에서는 납득할 수 없는 매우 긴 시간이다.

그림 4-14 순서대로 0.5ms + 0.1ms + 0.5ms가 걸린다고 가정

4.6 | 게임 서버의 품질

- 게임 서버의 성능을 높이는 법 : 서버의 단위 처리 속도를 높이기

서버의 단위 처리 속도를 높이려면, 프로그램이 더 빠르게 실행할 수 있게 코드 최적화나 알고리즘 최적화를 실행

RPG 게임에서는 몬스터나 플레이어 외 캐릭터(Non Player Character, NPC)의 길찾기 알고리즘(path finding)이 소요하는 처리 시간을 $O(1)$ 로 개선하고자 path table 테크닉을 쓰는 것도 최적화 방법 중 하나.

- 게임 서버의 성능을 높이는 법 : 서버의 과부하 영역을 분산

코드 프로파일링(code profiling) 이용하기 : 어떤 함수가 처리 시간을 많이 차지하는지 발견한 후 그것에 집중해서 성능을 개선하함.

함수 A의 처리 속도를 더 높일 수 있는 방법이 더 이상 없고, 그렇다고 함수 A를 실행하는 빈도를 낮출 수 있는 방법도 없다면 분산이 필요하다.

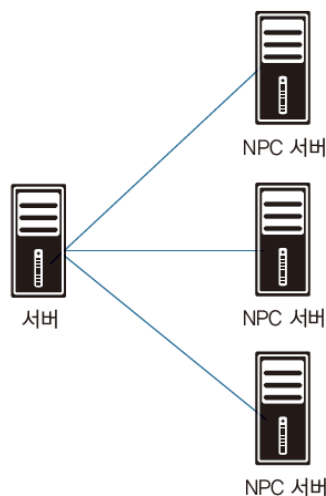


그림 4-15 NPC의 길찾기 알고리즘 분산

4.6 | 게임 서버의 품질

- 게임 서버의 성능을 높이는 법 : 네트워크 프로토콜 최적화

메시지의 양을 줄이기 : 양 자체를 줄이거나 압축하기



그림 4-16 네트워크 최적화 프로토콜 1: 양자화

메시지 교환 횟수 줄이기

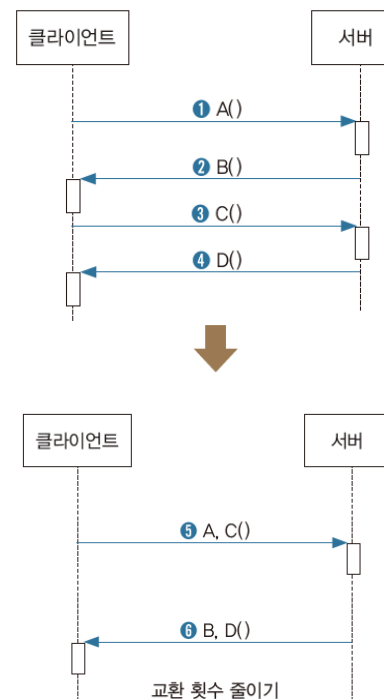


그림 4-16 네트워크 최적화 프로토콜 2
: 메시지 교환 횟수 줄이기

4.6 | 게임 서버의 품질

- 게임 서버의 성능을 높이는 법 : 네트워크 전송 시간 줄이기

고품질 네트워크 회선을 가진 데이터센터에 서버를 설치하기.

지리적으로 가까운 데이터센터에 서버들을 분산해서 설치하기.

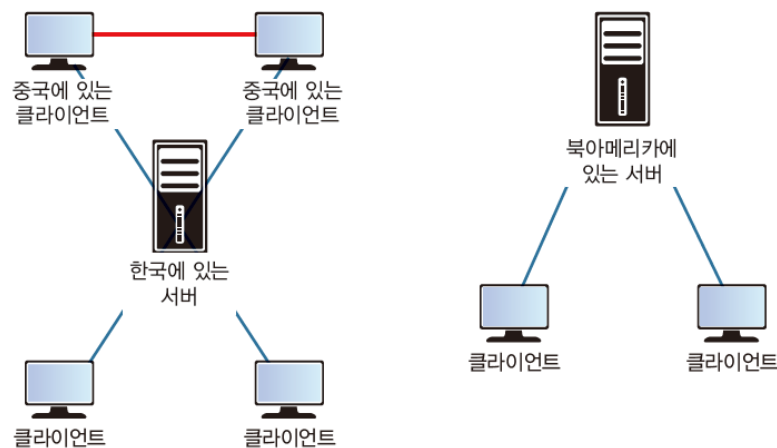


그림 4-18 지리적으로 가까운 데이터센터에 서버 분산

- 게임 서버의 성능을 높이는 법 : 서버를 거치지 않고 클라이언트끼리 직접 통신하게 하기

P2P(peer-to-peer) 네트워킹 : 클라이언트끼리 직접 메시지를 주고받는 것.

P2P 네트워킹은 서버에 걸리는 부담도 줄여 준다는 효과가 있으므로 파일 전송이나 음성 및 화상 채팅처럼 클라이언트끼리 주고받는 데이터의 양이 클 때 더욱 효과적임.

4.6 | 게임 서버의 품질

4.6.4 관리 편의성

데몬(daemon)/서비스(service) : 운영체제에 등록된 백그라운드 프로그램.

데몬은 프로그램 종료 같은 극히 제한적인 종류의 명령만 처리할 수 있으며, 백그라운드로 작동하는 프로그램이기 때문에 GUI는 물론이고 콘솔에 텍스트를 출력하면 화면에 아무것도 나타나지 않는다.

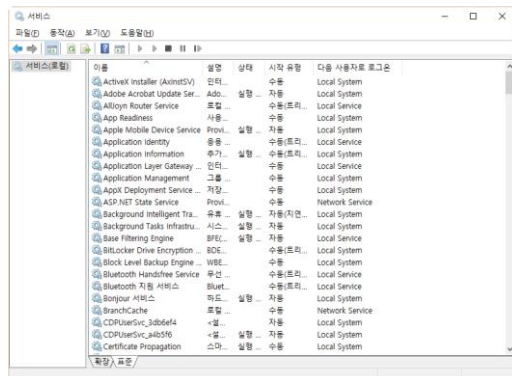


그림 4-19 윈도우에서 서비스들

관리 도구의 역할

서버 켜기/끄기
동시접속자 수 보기
CPU, RAM 사용량 보기



그림 4-20 서버 프로그램과 통신하는 관리 도구

4.7 | 플레이어 정보의 저장

- 온라인 게임에서 플레이어의 데이터 관리 중요성은 점점 커지고 있다.

보통 싱글플레이 게임의 플레이어 정보는 게임을 구동하는 컴퓨터 자체의 디스크에 저장된다.

온라인 게임에서 플레이어 정보를 클라이언트에 저장하지 않는 이유

1. 해킹에 취약하다.
2. 같은 사용자가 다른 기기를 사용할 때 문제가 된다.

그래서 대다수 온라인 게임은 플레이어 정보를 클라이언트가 아니라 서버에 저장한다.

플레이어의 데이터는 게임 서버에 직접 저장하는 경우도 있지만, 게임 서비스의 규모가 큰 경우에는 게임 서버의 디스크에 직접 저장하지 않고 별도의 서버 머신을 쓰기도 한다. 그리고 그 별도의 머신은 데이터베이스라는 소프트웨어 안에서 플레이어 데이터를 관리한다.



그림 4-1 규모가 클 때는 데이터베이스 머신을 따로 둬

4.7 | 플레이어 정보의 저장

• 데이터베이스 소프트웨어를 이용해서 저장하는 이유

1. 데이터 관리와 분석을 빠르게 할 수 있다.
2. 강력한 데이터 복원 기능이 있다.
3. '전부 아니면 전무'로 데이터를 변경할 수 있다.
4. 데이터 일관성을 유지시켜 준다.
5. 처리가 2개 이상 동시에 실행될 때 한 데이터가 동시에 여러 데이터를 액세스하면서 이상한 결과가 나오는 문제를 막아 주는 기능이 있다.
6. 장애에 대한 내성이 강하다.

표 4-2 단순 파일 시스템과 데이터베이스 장단점

구분	단순 파일	데이터베이스	비고
소프트웨어 비용	없다.	없거나 높다.	오픈 소스 제품은 제한적으로 무료다.
저장 및 로딩 속도	빠르다.	느리다	데이터베이스도 결국 파일 시스템을 사용한다.
데이터 관리, 분석 속도	느리다.	빠르다.	데이터베이스는 빠른 검색을 위한 인덱스 기능이 존재한다
데이터 백업, 복원 기능	없다.	있다.	-
원자성(atomicity)	불가능하다.	가능하다.	· 데이터 2개 이상을 all 또는 nothing으로 변경한다. · 데이터베이스의 트랜잭션 기능이다.
일관성(consistency)	없다	있다.	· 잘못된 상태의 데이터를 원천 봉쇄한다. · 데이터베이스의 constraints 기능이다.
고립성(isolation)	없다	있다.	· 경쟁 상태(data race)에서 자유로울 수 있게 하는 기능 이다. · 데이터베이스의 lock 기능이다.
지속성(duration)	없다.	있다.	· 장애 직전 상태로 복구할 수 있는지에 관한 것이다. · 데이터베이스의 로그 버퍼 기능이다

4.8 | 과제 대전 참가

- 10월 16일 수요일 수업은 과제 대전 참가로 대체
- 참가후 사진을 찍어서 email로 제출
 - 제목 [2019 게임서버프로그래밍 과제대전 참가 학번 이름]
 - 10월 17일까지 제출

4.8 | 서버 구동 환경

• 서버의 운영체제와 요구사항

서버를 구동하는 운영체제 : 리눅스와 윈도 서버가 골고루 쓰임

서버를 설치 할 때 요구사항 : 1. 인터넷 품질이 신뢰성이 있어야 한다. 인터넷이 중간에 끊어지거나 느려지지 않아야 한다.
2. 서버가 쉽게 고장 나지 않아야 한다. 컴퓨터를 365일 24시간 계속 켜 놓는 상황에서 장기간 고장이 없어야 한다.
3. 도난이나 침입 사고에서 안전해야 한다.
→ 이 점 때문에 규모가 큰 서버는 '데이터센터'라는 특별한 건물에 설치되는 것이 일반적이다.

• 클라우드 서버

클라우드 서버 : 리얼머신 위에 구동하는 가상 머신의 집합. 물리적 머신 한 대 안에서 여러 서버 운영체제가 동시에 작동함.



그림 4-23 온프림(물리적 자체) 서버와 클라우드 서버

4.8 | 서버 구동 환경

- 물리적 서버와 클라우드 서버

표 4-3 물리적 서버와 클라우드 서버 비교

구분	물리적 서버	클라우드 서버
특징	서버 컴퓨터 한 대에 운영체제 하나를 구동한다	서버 컴퓨터 한 대에 여러 운영체제를 가상 머신으로 구동한다.
서버 한 대당 유지 비용	낮다.	높다.
서버 증설 속도	낮다(하루).	높다(몇 분).
서버 철거 속도	낮다(경우에 따라 불가능).	높다(몇 분).
자동 스케일 아웃	불가능하다.	가능하다.
처리 속도의 균일성	높다.	낮다.
장애 처리	느리다.	빠르다.

4.8 | 서버 구동 환경

- 클라우드 서버가 제공하는 기능

IaaS(Infrastructure as a Service) : 가상 머신 자체를 제공하는 서비스. IaaS에서 생성한 가상 머신 안의 운영체제(시스템 레지스트리 등)는 마음대로 제어할 수 있다.

PaaS(Platform as a Service) : IaaS보다는 상위 계층에서 작동. 운영체제뿐만 아니라 운영체제 위에서 어떤 프레임워크 소프트웨어가 이미 구동되고 있으며, 이 프레임워크 위에 서버 코드나 데이터 파일을 업로드해야 한다.

SaaS(Software as a Service) : PaaS보다 상위에 있다. 코딩 자체가 불필요하며, 과금이 나 데이터 분석, 페이스북 로그인 연동 같은 특화된 기능들을 제공함.



IaaS

PaaS

SaaS

그림 4-24 계층에 따른 클라우드 서버 종류

4.9 | 서버 개발 지침

- 언어, 운영체제, 개발 도구 등 어떤 것을 선택하든 자신이 가장 잘 다루는 것을 쓰길 적극 권장함.
- 다양한 문제를 빠르게 해결하면 서버 개발에 사용한 언어, 운영체제, 도구 등이 자신에게 익숙해야 한다.
- 연구는 진보적으로 하고, 필드 적용은 보수적으로 할 것.

4.10 | 더 읽을 거리



- <http://goo.gl/EAbvUF>