소규모 팀으로 멀티플레이어 게임 개발하기

김진욱
(jinuk.kim@ifunfactory.com)

KGC 2018
-The Way To Survive-

01 들어가며

1. 리니지M
NCSOFT Corporation

2. 리니지2 레볼루션
Netmarble

3. 뮤오리진2
We[illegible]c.

4. 검은사막 모바일
[illegible]ABYSS

5. 다크에덴M
Entermate

6. 피망 포커: 카지노
NEOWIZ corp

7. 모두의마블 for ka
Netmarble

8. 세븐나이츠 for Ka
Netmarble

9. 오크: 전쟁의 서막
Efun Company

10. 에픽세븐
Smilegate Megaport

시장의 주류는 멀티플레이어 게임

I FUN FACTORY

멀티플레이어 게임 개발

내부의 핵심 역량?
외부에서 얻어다 쓰기?

KGC 2018
-The Way To Survive-

02 멀티플레이어 게임 톺아보기

멀티플레이어 MO 게임의 요소

- 인증: 유저 로그인 처리
- 매치메이킹: 적절한 다른 플레이어와 연결
- 게임 플레이: 게임 내 핵심 플레이 메커니즘
- 상점: 캐릭터, 액세서리, 게임 내 아이템 등등 판매
- 랭킹: 리더보드 / 친구끼리 경쟁
- 소셜 요소: 채팅, 길드/클랜, …

싱글 플레이 게임이라면

- 로그인 / 유저 데이터 로딩
- 인게임 진입 로비 / 메인 화면
- 인게임
  - 플레이
  - 보상 지급
- 상점 / 메일 / 퀘스트 등의 컨텐츠 시스템
- 랭킹 / 길드 / …

멀티플레이 게임이라면

- 싱글 플레이 요소 (거의) 전부에 더해서
- 다른 유저와 같이 인게임에 모아서 집어넣기
- 인게임 플레이 (변형)
  - 네트워크 지연 시간을 숨기고
  - 클라이언트 게임 메커니즘을 서버에 복제
  - 복제한 게임 메커니즘을 다시 클라이언트/서버 사이에 동기화하기

일정 늘어나는걸 줄이고
인력 풀도 조금만 늘려서
만들 수 있는가?

03 다른 게임 서비스는 어떻게 만들었을까?

PlayerUnknown's Battleground

KGC 2018

-The Way To Survive-

KGC 2018

-The Way To Survive-

Overwatch

KGC 2018

-The Way To Survive-

더 많은 개발자 / 더 다양한 스킬셋이 필요합니다

- 클라이언트/서버 간 통신: 네트워크 스킬
- 지연 시간 기획: 클라이언트/서버 간 지연 속이기
- 서비스: 운영을 위한 인프라/클라우드 다루기

I FUN FACTORY

적은 인원으로 문제 해결하기:
아웃소싱

KGC 2018
-The Way To Survive-

팀의 **핵심 역량을 구분**하고
그렇지 않은 것은 **외부에 맡기기**

아웃소싱

- 개발사가 모든 작업을 직접하진 않는다
- 외부에서 가져다 쓰는게 싸고 / 핵심 부분이 아니라고 판단하면 가져다 쓴다

아웃소싱: 게임 클라이언트 엔진

- 클라이언트 엔진을 인하우스로 개발하는 경우는 흔치 않다
- PC / 콘솔 / 모바일을 가리지 않고 외부 엔진 도입
- 대형 개발사도 UE4 같은 제3자 엔진 이용

아웃소싱: 서비스 인프라 스트럭처

- 많은 부분을 클라우드 서비스로 이용
- 특히, **IaaS** 가 대두
- 클라우드 공급자들이 한국 내 주요 고객으로 항상 게임 회사를 언급

아웃소싱: 게임 통계 분석

- 상당 수의 게임이 외부 서비스에 클라이언트 funnel 분석을 위한 데이터를 쌓는다
- 데이터를 직접 쌓은 경우에도, Google Analytics 나 BigQuery, AWS Elastic MR 같은 외부 분석 도구를 활용한다

예시 게임들의 추가 공통점:

데디케이티드 서버 활용
매치메이킹 기반

04 데디케이티드 서버

KGC 2018

-The Way To Survive-

클라이언트 기반 멀티플레이어

- DOOM 과 같은 게임은 클라이언트 하나가 (주로) 판정을 담당 (super peer)
- Super-peer 와의 연결이 단절된 경우 처리가 복잡하다 (모바일 네트워크의 흔한 문제)
- P2P 네트워킹은 모바일 환경에서 제대로 동작하지 않는다 (모바일 캐리어 망 = 거대한 공유기)

데디케이티드 서버의 등장

- 해당 판정만 처리하는 별도의 클라이언트 분리
  처음엔 **headless client** 라고 부름 (렌더링을 안해서)
- 별도로 띄우는 서버라서 **dedicated server** 라고 부르기 시작
- Quake, Counter Strike, RTCW, …

데디케이티드 서버

- 클라이언트 수정 최소화: 하나의 클라이언트에서 판정하는 것과 크게 다르지 않음
- 판정하는 클라이언트의 변조를 방지: 운영사가 직접 서버를 띄움
- **서버 전용 로직 없이도 멀티플레이 구현:** 필요한 서버 프로그래머 수를 적게 유지할 수 있다

데디케이티드 서버 개발 ≃ 클라이언트 개발

- 서버 시작 시간 단축 ≃ 클라 초기 로딩 시간 단축
- 메모리 최적화 ≃ 객체, 텍스처 최적화
- 시뮬레이션 변경 ≃ 클라이언트 틱 변경
- 로직 동기화 문제 ≃ 클라이언트 내 객체 동기화
- …

I FUN FACTORY

Life is not that easy

KGC 2018

-The Way To Survive-

서비스 확장하기 (1)

- 데디케이티드 서버:

  클라이언트 수준으로 메모리 사용

- 하나의 VM 위에서 수백-수천개 (=수백-수천 유저) 처리가 어려움

- 8vCPU 32G 메모리 VM에서 **수 개 - 수십개** 구동 하는게 한계

서비스 확장하기 (2)

- VM / 서버 수를 늘리는 확장 전략
- IaaS 클라우드의 API를 활용해서 데디케이티드 서버를 띄울 VM 수를 늘리고 / 줄이는 방법 사용

클라이언트 엔진의 한계

- 엔진 자체의 문제를 수정하는 복잡성
- Unreal Engine 제작사 EPIC은 Fortnite 개발 중에 UE4 수정
  - 배틀 로얄 게임이라 초기 인원 수가 일반적인 경우보다 훨씬 많음
  - 게임 시작 시점에 강하할 때 시야가 길어서 생기는 문제

문제: MO에서 MMO로 넘어가면

- 클라이언트 기술 기반: 엄청나게 많은 유저 (MMO) 동시 처리가 쉽지 않음
- **최적화 관점이 다르다**
  - 서버는 클라이언트 연결마다 쓸 수 있는 CPU/메모리 등을 **미리 분배**해서 설계
  - 클라이언트는 상대적으로 안정적인 렌더링 속도와 유저 입력 지연 시간에 촛점

05 매치메이킹

매치메이킹

- 비슷한 수준의 다른 플레이어와 게임하게 하기
- 플랫폼 별로 제공되는 기능을 쓰거나 직접 만들기

플랫폼 홀더가 제공하는 기능 쓰기

- Google: OpenMatch
- AWS: FlexMatch
- Xbox Live: SmartMatch
- (매치메이킹이라고 부르긴 힘들지만) Steam Matchmaking

직접 만들기

- (단순히) 필요한 유저 수가 될 때까지 기다린 후 매치 생성
- 레벨 혹은 ELO 레이팅 값을 써서 비슷한 유저끼리 묶어서 처리하기
- 지역 별 지연 시간을 구하고 적당한 위치 찾기
- …

쉽지 않은 부분

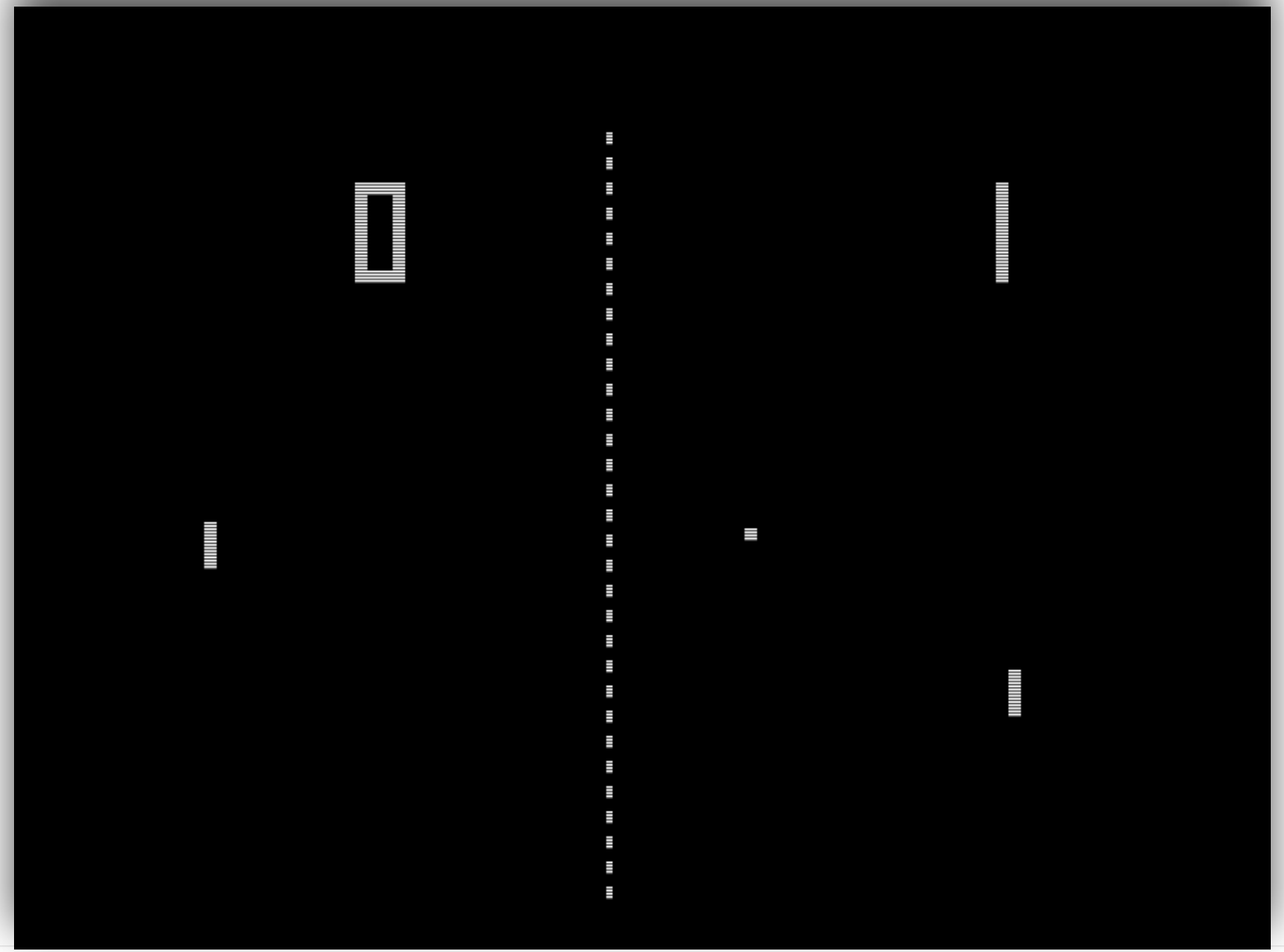
- 유저 수가 충분하지 않으면 매칭 자체가 어려움
- 유저 수가 많다면 적절한 유저를 매칭하는게 매우 복잡한 문제 ($_nC_k$: 조합론적 폭발)
- 유저 연결 관리: 네트워크 문제나 기다리다 지루해서 같은 이유로 유저가 수시로 매치메이킹 풀을 드나든다; 연결/유저/매치메이킹 큐 관리가 어려움

06 DEMO: 만들어보기

AWS + Unity3D + 아이펀엔진으로 만들어보기

- Pong 게임
- Unity3D 데디케이티드 서버
- 아이펀엔진 게임 서버: 인증/매치메이킹/서버관리
- AWS: 서버 용량 관리를 위한 API 활용

Pong 게임: ATARI의 고전 게임

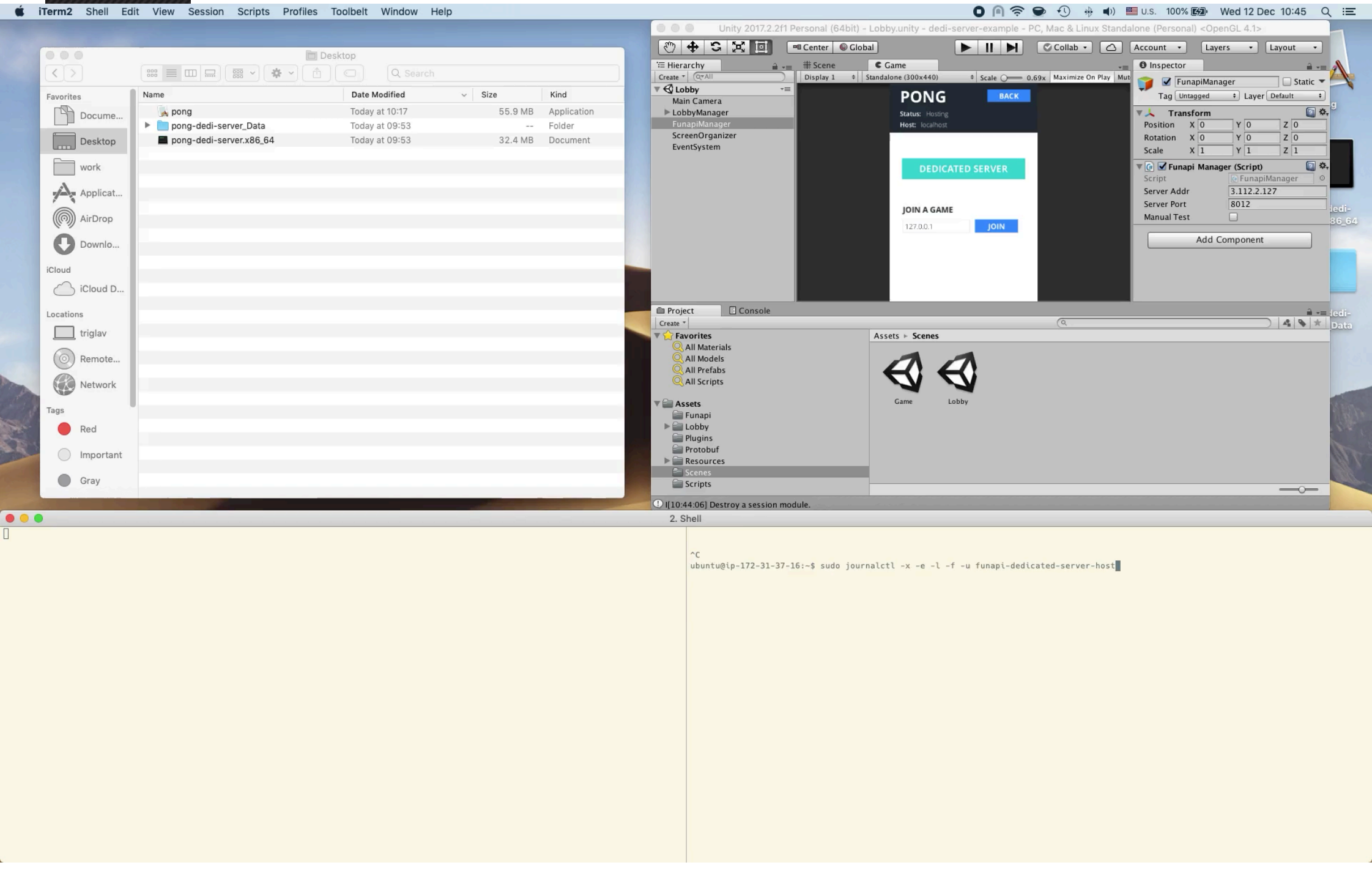


Unity 3D 데디케이티드 서버

- UE4 와 동일하진 않지만 데디케이티드 서버 제공
- UE4 보다는 좀 더 편하게 Windows / macOS / Linux 모두 서비스 플랫폼으로 이용할 수 있다

(광고) 아이펀 엔진

- **인증 / DB / 랭킹/ 데디케이티드 서버 관리 + α**
- ORM 을 활용해서 DB에 유저 데이터 기록
- 매치메이킹 기능을 써서 ELO 기반으로 매칭
- 매칭을 완료하면 VM 에서 데디케이티드 서버 실행
- 필요에 따라 EC2 VM을 지정한 지역에 띄우고,
  그 위에서 데디케이티드 서버 실행


iTerm2 Shell Edit View Session Scripts Profiles Toolbelt Window Help
Wed 12 Dec 10:45
Desktop
Favorites
Desktop
work
AirDrop
iCloud
Locations
triglav
Network
Tags
Red
Important
Gray
Name
Date Modified
Size
Kind
pong
Today at 10:17
55.9 MB
Application
pong-dedi-server_Data
Today at 09:53
Folder
pong-dedi-server.x86_64
32.4 MB
Document
Unity 2017.2.2f1 Personal (64bit) - Lobby.unity - dedi-server-example - PC, Mac & Linux Standalone (Personal) <OpenGL 4.1>
Hierarchy
Main Camera
LobbyManager
FunapiManager
ScreenOrganizer
EventSystem
Scene
Game
PONG
BACK
Status: Hosting
Host: localhost
DEDICATED SERVER
JOIN A GAME
127.0.0.1
JOIN
Inspector
FunapiManager
Transform
Position
Rotation
Scale
Funapi Manager (Script)
Server Addr
3.112.2.127
Server Port
8012
Manual Test
Add Component
Project
Console
Favorites
All Materials
All Models
All Prefabs
All Scripts
Assets
Funapi
Lobby
Plugins
Protobuf
Resources
Scenes
Scripts
Assets ▸ Scenes
Game
Lobby
I[10:44:06] Destroy a session module.
2. Shell
^C
ubuntu@ip-172-31-37-16:~$ sudo journalctl -x -e -l -f -u funapi-dedicated-server-host

샘플 코드 (1): 유저 인증

```
124 void OnLogin(const Ptr<Session> &session, const Json &msg) {
125   /* 메시지 형식
126    {"name": "blahblah"}
127    */
128
129   if (not msg.HasAttribute("name") or not msg["name"].IsString()) {
130     LOG(ERROR) << "OnLogin: name is not set";
131     return;
132   }
133   const std::string name = msg["name"].GetString();
134
135   AccountManager::CheckAndSetLoggedInAsync(
136       name,
137       session,
138       [](const string &name, const Ptr<Session> &session, bool success) {
139         if (not success) {
140           LOG(ERROR) << "User(" << name << "): failed to login.";
141           session->Close();
142           return;
143         }
144
145         OnLoginComplete(name, session);
146       });
147 }
```

샘플 코드 (2): 데이터 로딩 및 매치메이킹 시작

```
114  void OnLoginComplete(const string &name, const Ptr<Session> &session) {
115    Ptr<User> user = User::FetchById(name);
116    if (not user) {
117      user = User::Create(name);
118      user->SetRating(1400.0);
119    }
120    JoinMatchmaking(name, user->GetRating());
121  }
```

샘플 코드 (3): 매치메이킹 조건 검사

```
23  // `user` 를 `match` 에 넣을지 확인
24  bool CheckMatch(const MatchmakingServer::Player &user,
25                  const MatchmakingServer::Match &match) {
26    // ELO 레이팅 차이가 100이하일 때만 매치 허용
27    double r1 = match.players[0].context["user_data"]["rating"].GetDouble();
28    double r2 = user.context["user_data"]["rating"].GetDouble();
29    double gap = r1 - r2;
30    return gap < 100.0 and -100.0 < gap;
31  }
32
33  // "JoinMatch" 이후에 호출
34  MatchmakingServer::MatchState CheckCompletion(
35      const MatchmakingServer::Match &match) {
36    if (match.players.size() == 2) {
37      return MatchmakingServer::kMatchComplete;
38    }
39    // We need more players to make a match.
40    return MatchmakingServer::kMatchNeedMorePlayer;
41  }
```

샘플 코드 (4): 데디케이티드 서버 생성

```
102   std::vector<std::string> dedicated_server_args;
103
104   std::vector<std::string> user_list;
105   std::vector<fun::Json> user_data;
106   for (const auto &user : match.players) {
107     fun::Json user_json;
108     user_json.SetObject();
109     user_json["x"] = "y";  // dummy
110     user_json["id"] = user.id;
111     user_list.push_back(user.id);
112     user_data.emplace_back(user_json);
113     match_data["Users"].PushBack(user_json);
114   }
115
116   DLOG(INFO) << "Spawning Match(" << match.match_id…
119   // Just send users to the dedicated server.
120   // (Assume that the "id" is corresponding to the account id of the user.)
121   fun::DedicatedServerManager::Spawn(
122       match.match_id,
123       match_data,
124       dedicated_server_args,
125       user_list,
126       user_data,
127       OnDedicatedServerSpawned);
128 }
```

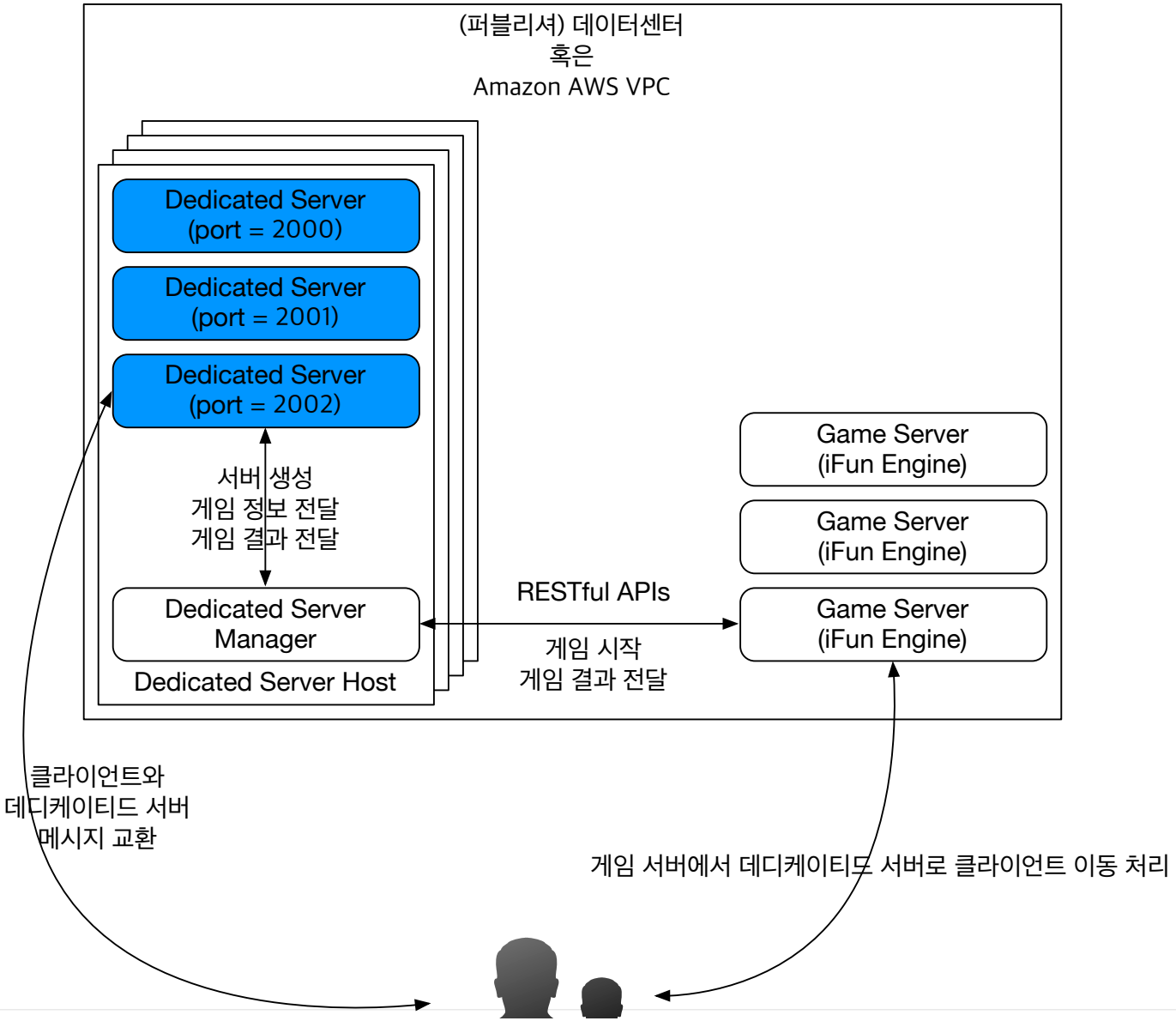
샘플 코드 (5): 게임 결과 후처리 1/2

```
59  void OnMatchResultReceived(const fun::Uuid &match_id,
60                             const fun::Json &data,
61                             bool success) {
62    if (not success) {…
65    }
66
67    LOG(INFO) << "Match(" << match_id << "): finished with result: "…
70    if (not data.HasAttribute("result", fun::Json::kObject)…
75    }
76
77    const std::string winner = data["result"]["winner_uid"].GetString();
78    const std::string loser = data["result"]["loser_uid"].GetString();
79
80    fun::Event::Invoke([winner, loser]() {
81      // 유저 데이터를 가져와서 ELO rating 업데이트
82      Ptr<User> win_user = User::FetchById(winner);
83      Ptr<User> lose_user = User::FetchById(loser);
84      if (not win_user || not lose_user) {
85        LOG_IF(ERROR, not win_user) << "User(" << winner << ") not found.";
86        LOG_IF(ERROR, not lose_user) << "User(" << loser << ") not found.";
87        return;
88      }
89
```

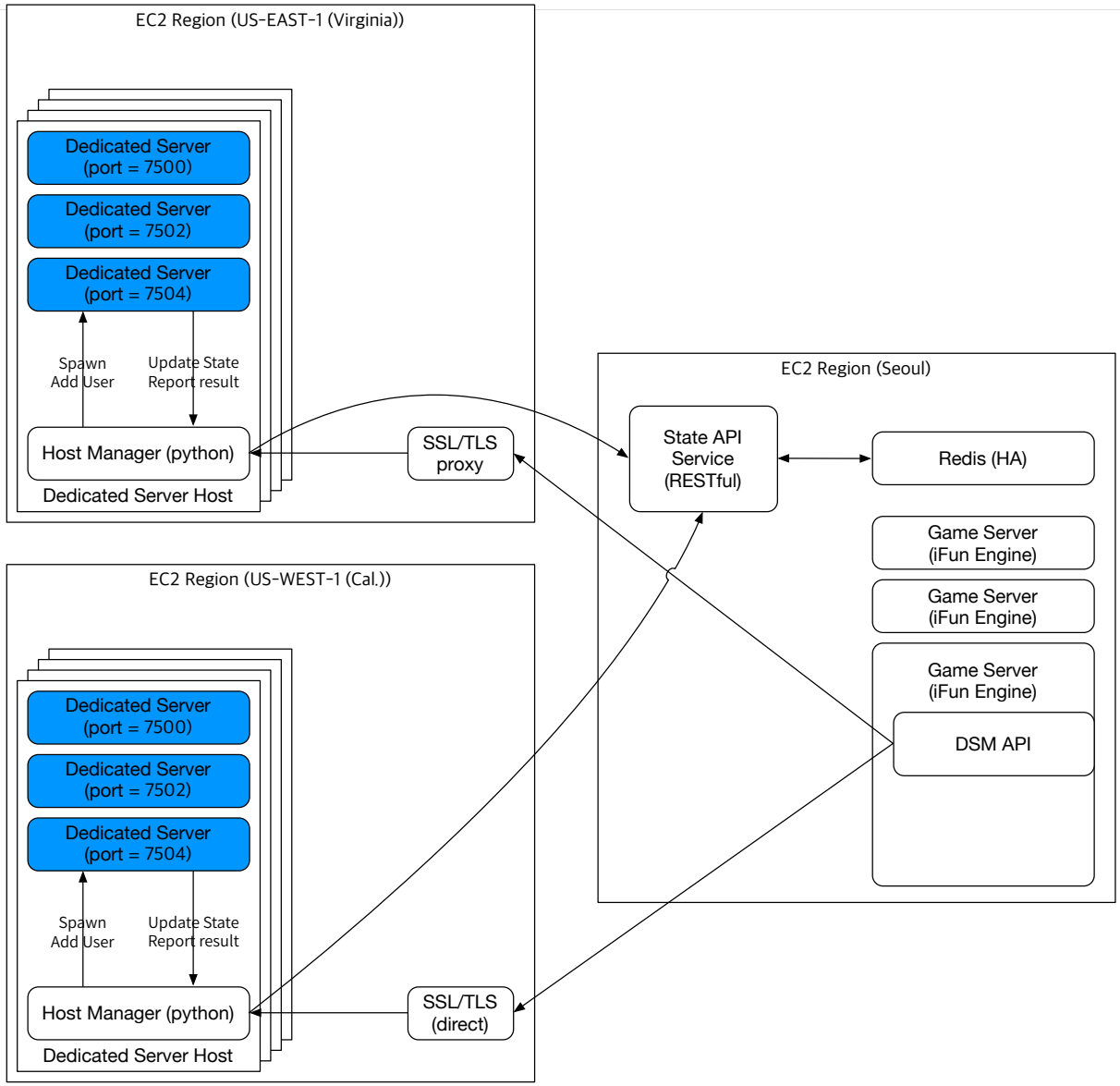
샘플 코드 (6): 게임 결과 후처리 2/2

```
90          // ELO rating 재계산
91          double r1 = win_user->GetRating();
92          double r2 = lose_user->GetRating();
93
94          double R1 = std::pow(10.0, r1 / 400.0);
95          double R2 = std::pow(10.0, r2 / 400.0);
96
97          double E1 = R1 / (R1 + R2);
98          double E2 = R2 / (R1 + R2);
99
100         double r1_new = r1 + 32.0 * (1.0 - E1);
101         double r2_new = r2 + -32.0 * E2;
102
103         win_user->SetRating(r1_new);
104         lose_user->SetRating(r2_new);
105
106         LOG(INFO) << "Winner(" << winner << "): rating changed from "
107                   << r1 << " -> " << r1_new;
108         LOG(INFO) << "Loser(" << loser << "): rating changed from "
109                   << r2 << " -> " << r2_new;
110     });
111 }
```

많은 수의 데디케이티드 서버로 확장하기



글로벌 서비스로 확장하기



I FUN FACTORY

마무리

KGC 2018

-The Way To Survive-

I FUN FACTORY

잘할 수 있는 일로 성공하기

KGC 2018

-The Way To Survive-

- 게임 개발은 여러가지 요소가 종합적으로 필요한 어려운 일
- 작은 팀으로 좋은 게임을 만드는게, 큰 팀으로 모든 기술적인 문제를 푸는 것보다 쉬울 수 있다
- 클라우드 서비스, 데디케이티드 서버처럼 외부 기술을 활용해서 게임 플레이에 집중한다면 상대적으로 빠른 시간 내에 런치가 가능하다

I FUN FACTORY

Q&A

KGC 2018

-The Way To Survive-

THANKS!

Great Technology For Great Games, iFunFactory

iFunFactory

jinuk.kim@ifunfactory.com

www.ifunfactory.com

+82-70-4923-6566

KGC 2018

-The Way To Survive-