PiggPARTY의 실시간 통신 시스템

최흥배
https://github.com/jacking75/choiHeungbae

PiggPARTYでのリアルタイム通信の仕組み

2015年05月14日(木)
テーマ:サービス・技術

ピグ事業部でサーバーサイドエンジニアをしている有馬です。

先日、弊社よりスマートフォン向けネイティブアプリとして、「PiggPARTY」がリリースされました。
PiggPARTY Androidアプリ
iOSアプリは近日公開予定です

http://ameblo.jp/principia-ca/entry-12016672808.html

PiggPARTY는 스마트 폰 앱에서 얼굴, 옷 등 여러 부품을 조합하여 원하는 대로 피그(아바타)를 만들 수 있다.

시부야 구역과 하라주쿠 지역 같은 현실을 본뜬 지역이나,
개인 취향의 가구로 변경한 자신의 방에서 파티(이벤트)를 개최하거나,
다른 사용자와 텍스트 채팅이나 스탬프 등으로 실시간으로 커뮤니케이션을 즐길 수 있는 서비스.

PiggPARTY-ピグパーティ

CyberAgent Inc. - 2015년 7월 7일 - ③

캐주얼 게임

설치 | 위시리스트에 추가

인앱 구매 제공

✕ 이 앱과 호환되는 기기가 없습니다.

★★★½☆ (1,236)

g+1 +106 Google에 이 URL 추천

인기 개발자

https://play.google.com/store/apps/details?id=jp.co.cyberagent.miami

개요

PiggPARTY는 동기적인 실시간 커뮤니케이션을 실현하기 위해서 새롭게 실시간 통신 서버 라이브러리를 밑 단계부터 개발하고 있으며, 이번에는 그 구조에 대해서 소개한다.

또한, 회사 내에서는 그 라이브러리를 **toychat**라는 명칭으로 개발하고 있으며, 이후 글 중에서도 이 명칭을 사용한다.

(PiggPARTY에서는 다른 것도 clay(클레이 애니메이션), origami(종이)라는 독특한 명칭으로 라이브러리 개발이 이루어지고 있다.)

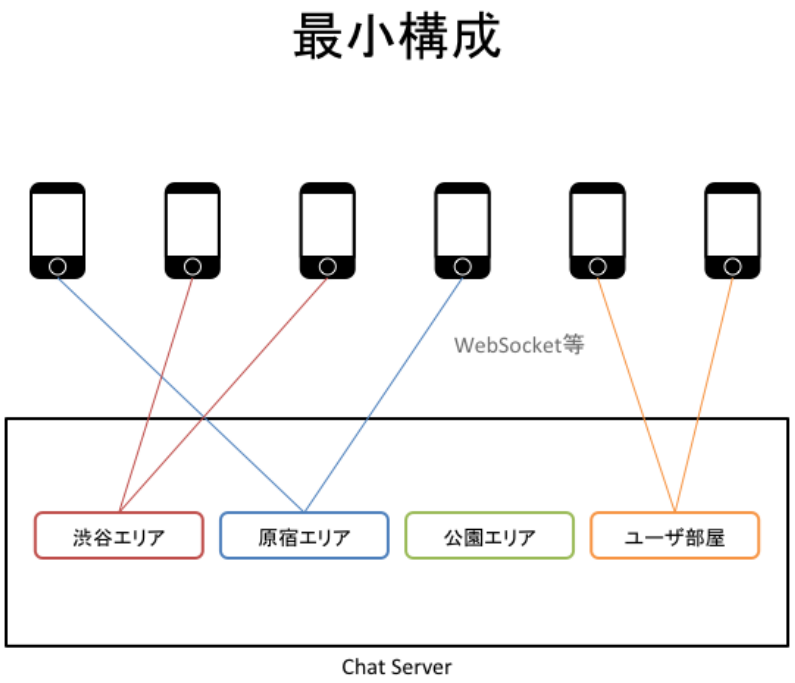
toychat의 특징

toychat은 **Node.js로 만든 서버 라이브러리**이며 실시간 채팅 어플리케이션이나 온라인 게임의 게임 서버 등에서 이용하는 것을 계획하고 만들었다.

특징은 아래와 같다.
- TCP 기반의 분산 실시간 메시징 서버
- MQTT와 WebSocket 등 멀티 프로토콜을 선택 가능
- 페일 오버

toychat의 구조

분산 리얼 타임 메시징
TCP 베이스에서 채팅 같은 실시간 메시징 서버를 생각할 경우 최소 구성은 다음과 같을 것이다.

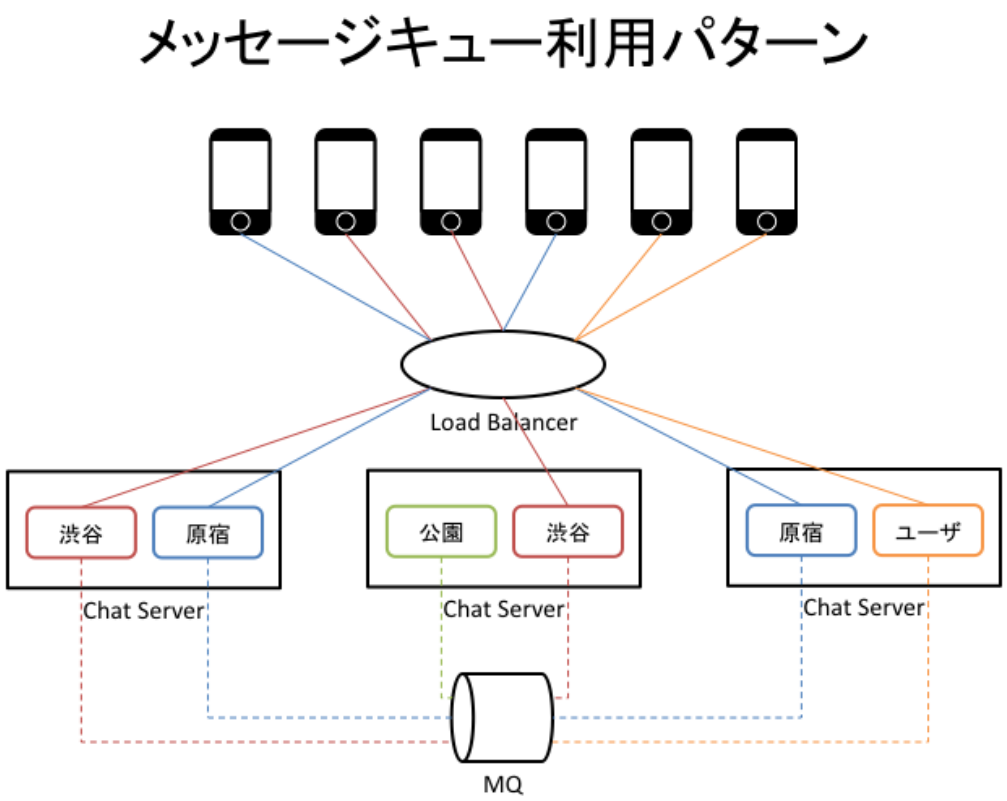


1대의 서버, 사용자 사이를 WebSocket 등 서버에서 Push 통신이 가능한 프로토콜로 접속하고 서버는 그 지역에 송신된 메시지를 그 지역에 접속 중인 사용자들에게 전달한다.

그러나 **서버 1대로 대량의 사용자 부하를 다루기에는 한계가 있다. 그래서 서버를 분산하지만 서버 간에 동일 영역에 대한 메시지를 어떻게 동기화 하느냐가 문제가 된다.**

사내 외 사례에서 몇 가지 패턴을 참고 했다

RabbitMQ, Redis등 메시지 큐를 이용한 패턴



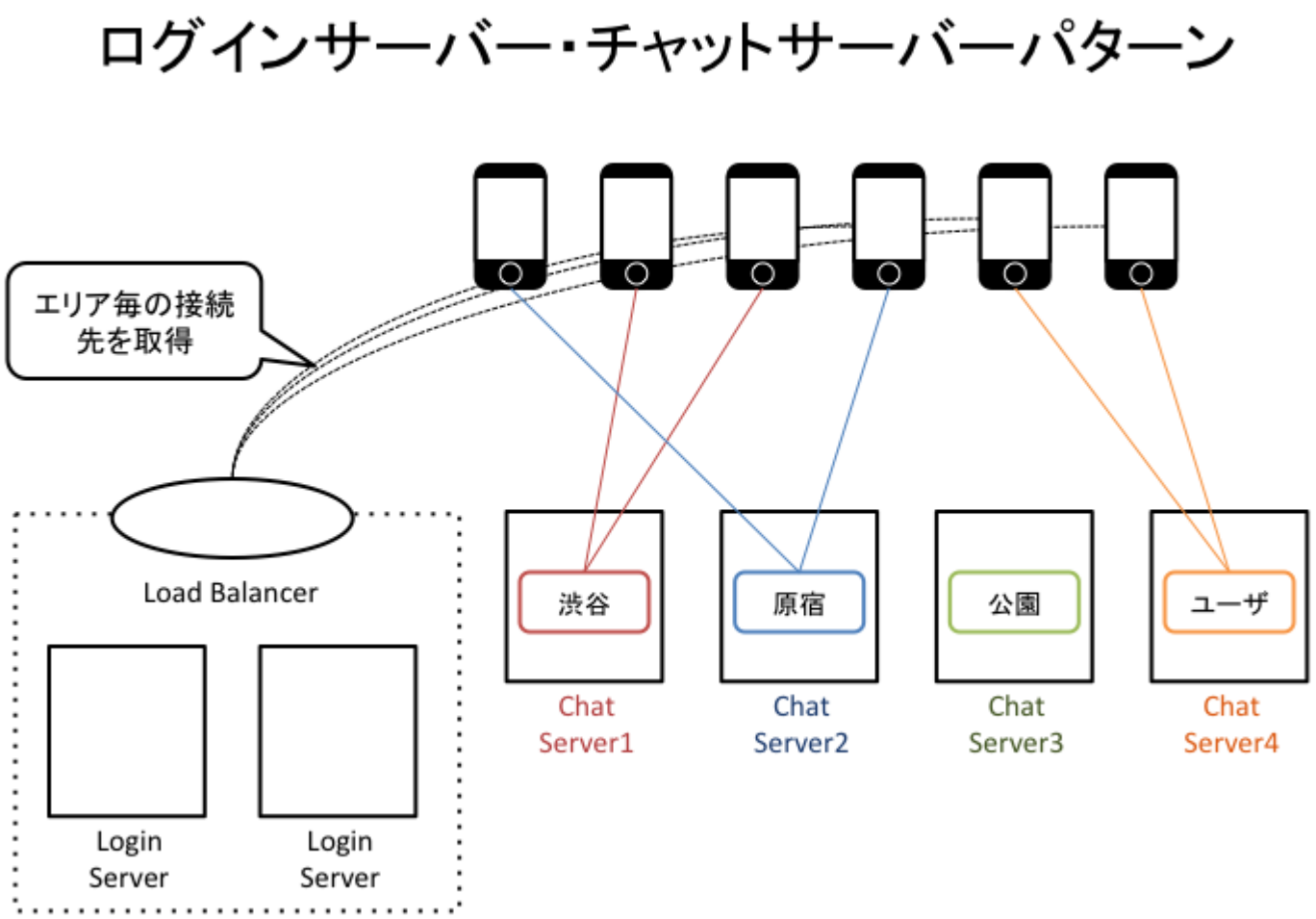
이 구성에서는 지역에 관계 없이 사용자의 접속은 로드 밸런서에 의해 균등하게 편성된다.

백엔드에 메시지 큐를 배치하고 큐를 통해서 서버 간에 메시지를 교환함으로써, 동일 영역에 메시지를 동기화 한다.
분산 채팅 시스템에서는 기본적인 구성이다.

매우 심플하고 stateless로 스케일 하기 쉽지만, PiggPARTY처럼 메시지 송신만 아니라 지역에 관한 정보를 많이 취급하는 등의 서비스에서는 동일 지역의 정보를 동기화하기 위한 데이터 스토어가 필요하게 되고, 애플리케이션 코드가 복잡화하기 쉽다.

예를 들어 PiggPARTY에서는 지역 내에서의 사용자의 좌표 정보를 사용자가 이동할 때마다 갱신하는데, 이것들을 복수의 서버로 공유 갱신하게 되면, 갱신 부하와 병렬 갱신의 동기화 등 보통 수단으로는 하기 어렵다.

로그인 서버, 채팅 서버의 패턴



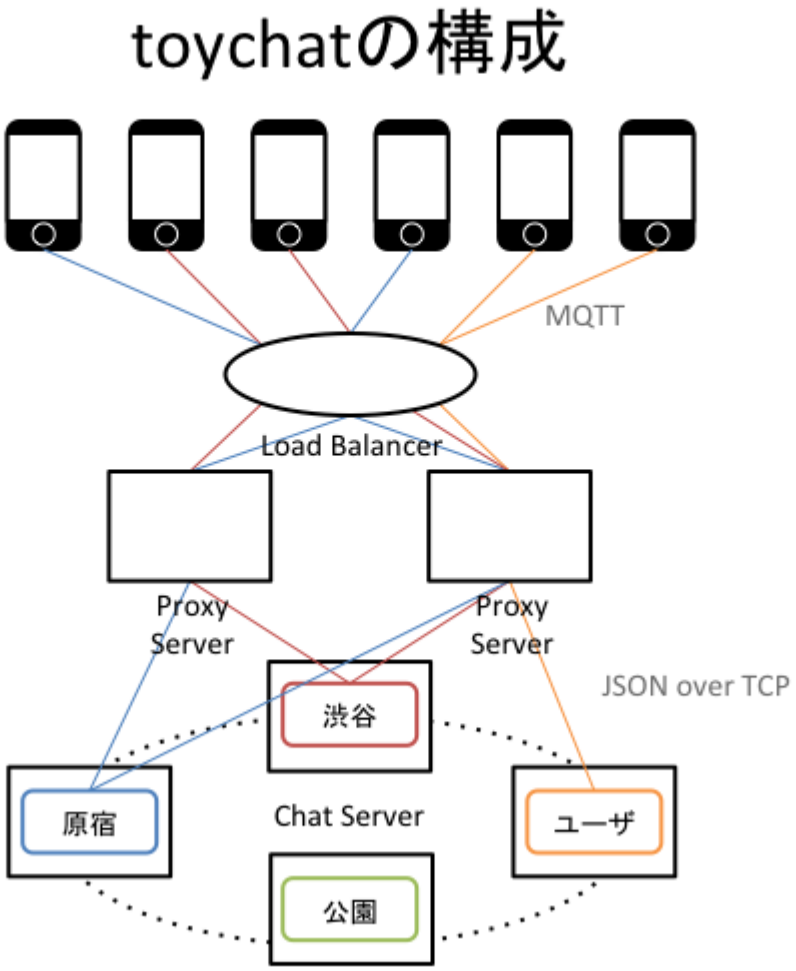
이 구성에서는 사용자는 지역에 입주 할때 어떤 채팅 서버에 접속하면 좋을지를 로그인 서버에 문의, 지시된 접속처로 접속한다.

동일 에리어 접속은 반드시 같은 서버로 분산되므로 지역에 관한 정보를 서버의 온 메모리에서 다룰 수 있는 고속이며 비교적 간단하게 애플리케이션을 구축할 수 있다.

PC 아메바 피그는 이것과 비슷한 구성으로 되어 있다.

채팅 서버를 병렬로 늘어놓기로 스케일 할 수 있지만, 채팅 서버 수만큼 글로벌 IP가 소요되는 등 운용 면에서 약간 복잡하게 되는 경향이 있었다.

toychat의 구성


toychatの構成
MQTT
Load Balancer
Proxy
Server
Proxy
Server
JSON over TCP
渋谷
原宿
Chat Server
ユーザ
公園

피그 라이프 등 PC 피그 게임의 구조에 꽤 가까운 구성이며, 좀 더 역할을 명확, 간편하게 한 구성으로 되어 있다. 각각의 역할을 간단히 설명하겠습니다.

ProxyServer

사용자의 접속을 접수, 영역에 대한 메시지와 접속단 등의 이벤트를 지역에 대응하는 ChatServer 프로세스에 라우팅 한다. 또 ChatServer로부터 송신된 메시지를 이용자에게 송신한다.

ChatServer

ProxyServer를 통해서 유저와의 메시징을 한다.
각 ChatServer 프로세스는 각각 다른 하나 이상의 지역을 담당하고, 같은 지역에 있는 이용자에게 메시지를 송신하거나 지역 내로 메시지 방송 등을 한다. 또 인터널 통신으로 다른 지역을 담당하는 ChatServer 프로세스에 메시지를 발송할 수도 있다.

애플리케이션에서의 이용

ProxyServer와 ChatServer는 Node.js의 EventEmitter를 승계했고, 유저의 접속, 접속 단절, 영역에 대한 입퇴실, 메시지 송신 등 각 액션에 대한 이벤트를 생성한다.
라이브러리를 이용하는 애플리케이션은 리스너를 등록함으로써 각 이벤트 타이밍에서 임의의 처리를 실행할 수 있다.

다음은 라이브러리를 이용하는 코드의 이미지이다.
분산 환경에서도 단일 서버에서 처리하는 듯한 감각으로 개발할 수 있다는 이미지가 전달되기를 바란다.

```
/*
 * ChatServer에 송신된 메시지를
 * 지역에 입실 중의 다른 사용자에게 방송하는 코드 예
 *
 * chatUser 메시지 송신자를 나타내는 객체
 * room     룸(지역)를 나타내는 문자열
 * body     메시지
 */
chatServer.on("message", function(chatUser, room, body) {
  // 지역에 입실 중의 유저를 취득
  var joinedUsers = chatRooms[room];

  if (joinedUsers) {
    // 메시지의 발신자외를 취득
    var others = _.values(joinedUsers).filter(function(joined) {
      return joined.userId !== chatUser.userId;
    });

    //  보낸 사람 이외에 메시지를 브로드캐스트(실제로는 ProxyServer를 통해서 전송된다)
    chatServer.send(others, room, body);
  }
});
```

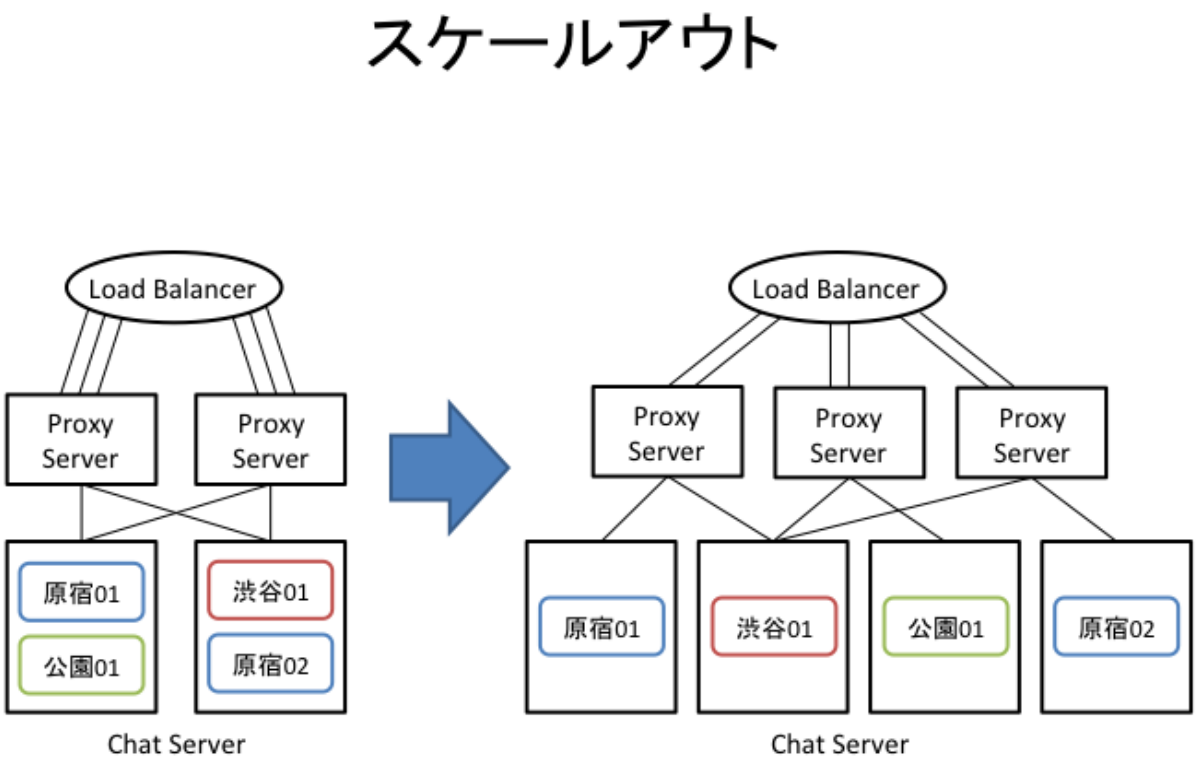
통신 프로토콜

toychat에서는 사용자 단말기와 통신 프로토콜은 교체 가능하게 만들었다.

예를 들어 PiggPARTY가 MQTT(※) 프로토콜을 사용하고 있지만 WebSocket 등 다른 프로토콜을 사용할 수도 있다.

ProxyServer와 ChatServer 간은 JSON over TCP로 sokcet 통신을 하고 있다.

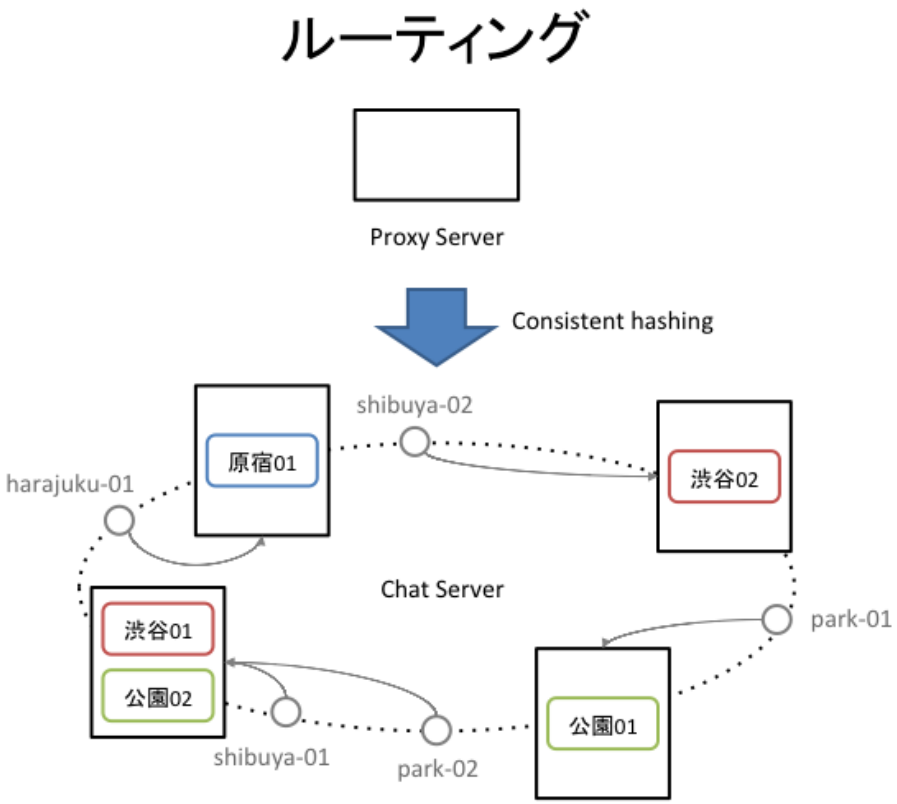
スケイル 아웃



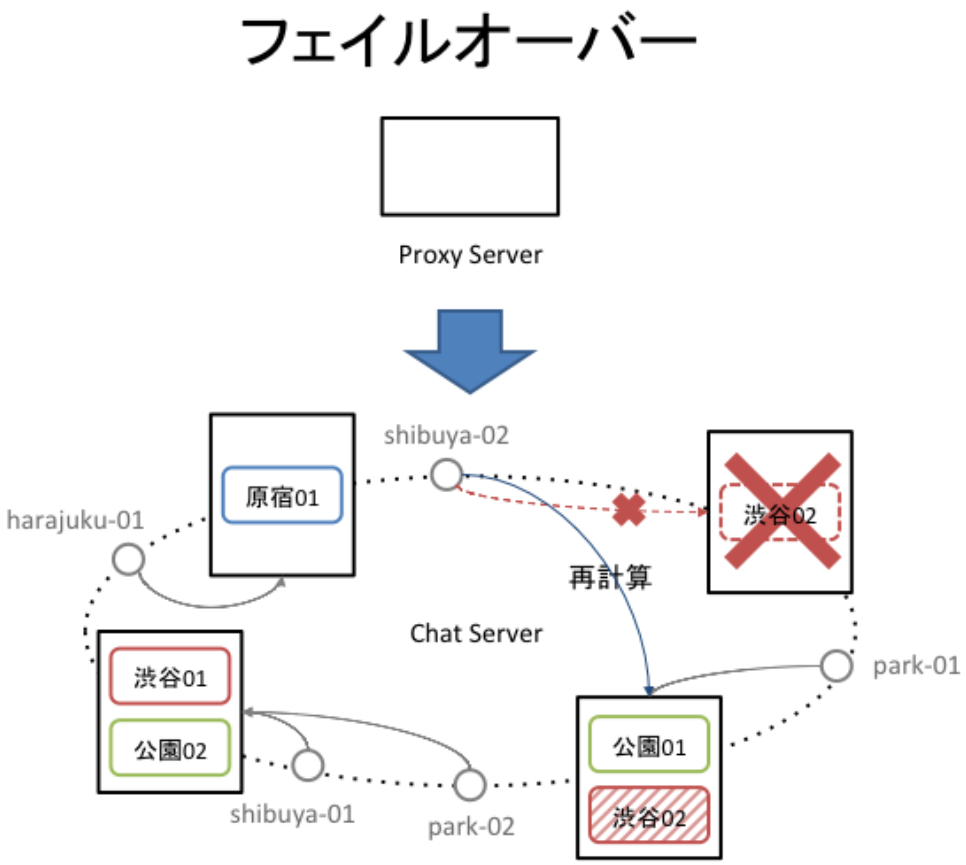
이처럼 서버를 옆으로 늘려가면서 ProxyServer, ChatServer 간의 접속 수가 TCP 포트 수의 상한을 넘지 않는 수준까지는 이 구성으로 부하를 분산할 수 있다.

라우팅과 페일 오버

각 ChatServer 프로세스가 어느 지역을 담당할지는 Consistent hashing을 사용하여 결정하고 있다.



ChatServer 프로세스에 장애가 발생한 경우는 해시 링 재계산을 실시하여, 지역 할당을 업데이트한다.



각 서버 간에 올바른 지역 할당을 실시하려면, 각 ChatServer 프로세스의 목록을 동기화 하고, 각 서버 간에 같은 해시 링을 계산할 필요가 있다.
프로세스 목록 동기에는 여러가지 방법을 취할 수 있지만 PiggPARTY에서는 Serf(※)을 이용하여 동기를 실시하고 있다.

프로세스를 감시하고 이상이 있었을 경우에는 Serf를 통해서 정보를 전파한 프로세스 목록 갱신을 한다.

Chat Serverプロセスリストの同期