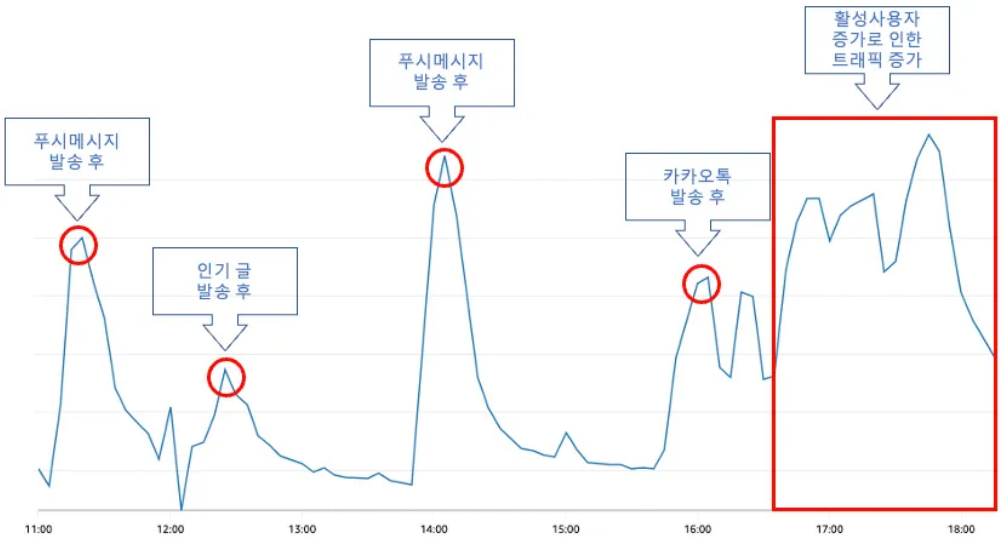
# 트래픽

| | |
|---|---|
| 주차 | 35주차 |
| 스터디 일자 | @2024/07/18 |

## 트래픽

> 트래픽(Traffic)이란, 네트워크 내부에 일정 시간 동안 흐르는 데이터 양을 뜻한다.
>
> 인터넷이나 블록체인 등과 같은 네트워크를 이용하기 위해서는 먼저 서버에 접속해야 하고, 그때마다 필요한 정보를 다운받아야 한다.
>
> 이 때, **네트워크를 타고 이동하는 데이터를** ***트래픽*** 이라고 한다.




## 대규모 트래픽 처리

**서버가 터졌다.**

⇒ *대규모의 트래픽이 순간적으로 서버에 몰림으로써 서버가 그 모든 요청을 다 수용하지 못하는 상황을 말한다.*

이러한 **대규모 트래픽 처리**가 중요한 이유는, 비슷한 서비스를 제공하는 경쟁자가 여럿 있는 시장속에서의 서비스 제공자라면  서버가 > **고객 이탈** < 이라는 손해로 이어질 수 있다.

👉 따라서 대규모 트래픽 처리는 경쟁사가 많은 업종 일수록 아주 중요하게 해결해야 하는 과제라고 할 수 있다.

# 서버가 터지는 이유?

***서버***는 결국 외부로부터 들어오는 요청을 받아 처리해주고 응답을 해주는 프로그램이 돌아가고 있는 어느 컴퓨터이고,

이 말은 즉, ***서버*** 또한 그 처리 속도와 한계가 `CPU`, `메모리`, `저장장치`에 영향을 받는다는 소리다.

## 메모리 오버플로우

서버가 모든 Task를 동시에 처리할 수는 없기 때문에 결국 Task들을 **큐**에 넣어 순차적으로 처리하게 되는데,

이 큐는 `메모리` 에 존재한다.

→ 즉, 요청을 들어오는 족족 큐에다가 넣다보면 **메모리 오버플로우**가 날 수 있다는 것!

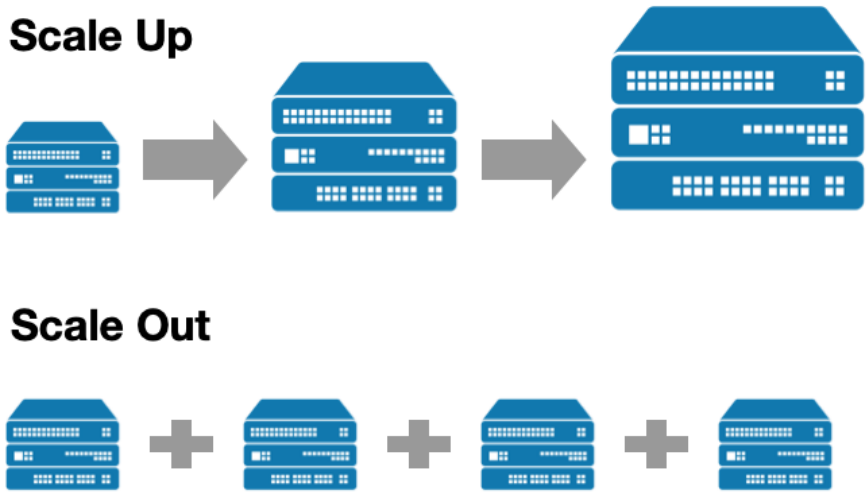
웹 서버라는 것은 결국 프로그램인데 메모리 오버플로우가 나면 종료가 되고, 그렇게 서버가 터지는 것이다.

이 외에는 큐 오버플로우, 타임아웃 문제가 있다.

# 서버를 지키는 법

💡 서버가 터지는 이유는 **결국 아직 처리하지 못한 요청이 쌓여서** 이기 때문에, 단순히 생각해 요청을 충분히 빠르게 처리하면 된다.

## 서버를 지키는 2가지 방법




### 1. Scale-out

처리하는 서버를 물리적으로 늘려버리는 방법

### 2. Scale-up

처리하는 서버를 더 강하게 만들어버리는 방법

## 구체적인 해결책

### 1. 기억장치 성능 향상

> 기억 장치의 종류, 그 자체를 변경할 수도 있다.

DB의 세계에는 `In-memory Datebase`라는 필살기와 같은 것이 존재한다.

- 인메모리 DB의 대표격으로는 `Redis`와 `Memcached`가 있다.
- 이 둘의 대표적인 사용처는 ***캐싱*** 인데,

  캐싱이 반복적으로 요청되는 **특정 데이터에 대한 *Read* 비용 감소**를 목적으로 하고 있다는 것을 생각하면 캐싱 또한, 대규모 트래픽 처리의 한 부분이라는 것을 알 수 있다.

하지만 캐싱만으로는 한계가 있다.

→ 왜냐하면 DB의 정보를 읽어오는 것만 빠르게 하기 때문이다.

예를 들어 선착순 발매 쿠폰의 개수처럼 계속해서 변경이 되어야 하는 정보라면, 변경된 개수는 계속해서 기록이 되어야 하는데,

이런저런 문제가 생겨서 `쓰기 동작`이 실행되기 전에 데이터가 날아가버리면, 이는 큰 대참사로 이어지게 된다.

👉 그렇게 때문에 서버는 **별개로 메모리를 사용하면서** 데이터를 DB처럼 조작할 수 있는 `In-memory-DB`가 필요하게 된다.

많은 회사들이 인메모리 DB 중에 **Redis**를 선호하는 이유에 이 영속성 처리가 한 몫을 한다.

**Redis**는 Replica를 통해 데이터를 복제해 Master에 장애가 발생해 사용이 불가능해진다고 해도 데이터 자체는 남게 되기 때문이다.

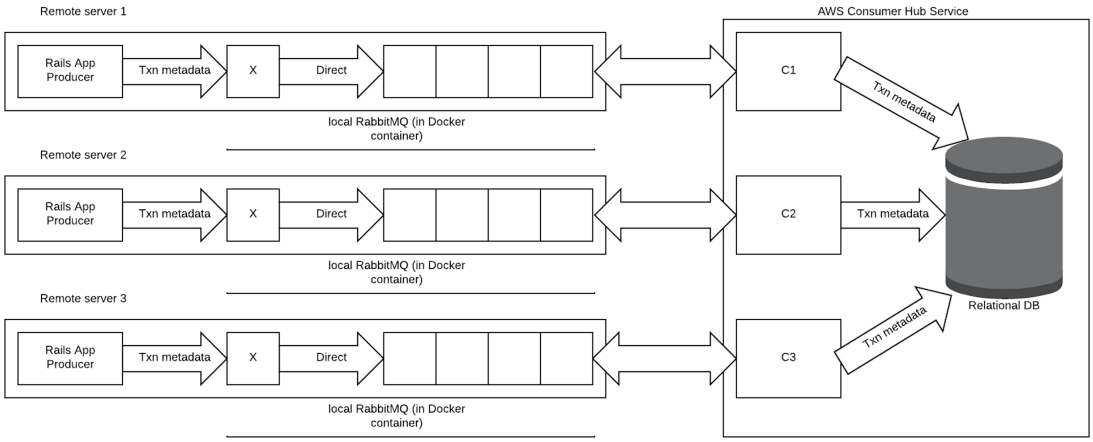
## 2. I/O 큐의 성질 변화

> 어느 회사에서는 이렇게 판단할 수도 있다.
>
> 아주 잠시만 트래픽이 폭주하는 것이라, 잠시 뒤에는 통상 트래픽으로 돌아올 것이기에 굳이 인메모리 DB까지는 필요하지 않을 것이다.

다른 방법으로는, **DB에 기록하는 부분**만 잠시 미루는 방법이 있다.

CPU와 메모리를 이용해 데이터를 처리하는 부분은 충분히 빠르기 때문이다.

→ 그럴 때 사용할 수 있는 게 `메세지 큐` 이고, 대표적인 메세지 큐에는 `RabbitMQ` 가 있다.




- 위와 같이 DB앞에 `MQ`를 두게 된다면, 사용자의 요청에 대한 응답은 DB에 실제로 넣지 않았지만, **넣어질 것이라고 가정한 상태에서 할 수 있게 된다.**
- 즉, 실제 저장이라는 하나의 스탭을 생략하고 응답을 하게 되므로 각 요청에 대한 응답 시간이 짧아지게 되는 것이고,
    - 이는 곧 총 처리 시간의 감소라고 볼 수 있게 되는 것이다.

✨ 총 처리 시간이 감소해, 요청이 들어오는 속도만큼 빨라지거나 이보다 더 빨라진다면, 요청이 쌓일 일이 없어져서 **서버가 안정적으로 동작하게 된다.**