4장 — 처리율 제한 장치의 설계

처리율 제한 장치

- 클라이언트, 서비스가 보내는 트래픽의 처리율(rate)을 제어하기 위한 장치

다른 면접과 마찬가지로 요구사항 먼저 확인

- 제한 장치의 위치
- 제어 규칙의 유연성
- 제어 기준 (API/ IP)

처리율 제어 장치를 어디에 둘 것인가?

- 클라이언트는 위변조가 가능
- 서버에 위치 가능
- 보통 게이트웨이에 구현

처리율 제한 알고리즘

- 토큰 버킷
  - 요청이 처리 될 때마다 토큰을 하나씩 사용하는 방법
  - 제어 인자 2개 → 버킷 크기, 토큰 공급률
- 누출 버킷
  - 큐에 빈자리가 있으면 요청 추가, 큐가 가득차 있을 경우 새 요청 버림
  - 지정 시간마다 요청을 꺼내어 처리 → 요청 처리율이 고정
  - FIFO Queue로 구현
  - 제어 인자 2개 → 버킷 크기, 토큰 공급률
- 고정 윈도 카운터
  - 타임라인을 고정된 간격으로 나눈 window 마다 카운터 관리

    - window 경계 부근에서 트래픽 몰림 가능
- 이동 윈도 로그
    - 고정 윈도 카운터의 문제를 해결하기 위해 나온 방법
    - 거부된 요청까지 로그로 관리하기 때문에 다량의 메모리 사용
- 이동 윈도 카운터
    - 고정 윈도 카운터, 이동 윈도 로그를 합쳐 보안한 방법
    - 현재 1분간의 요청 수 + 직전 1분간의 요청 수 * 이동 윈도와 직접 1분이 겹치는 비율
    - 짧은 시간 몰리는 트래픽 대응 가능, 메모리 효율 좋음

카운터 데이터 관리

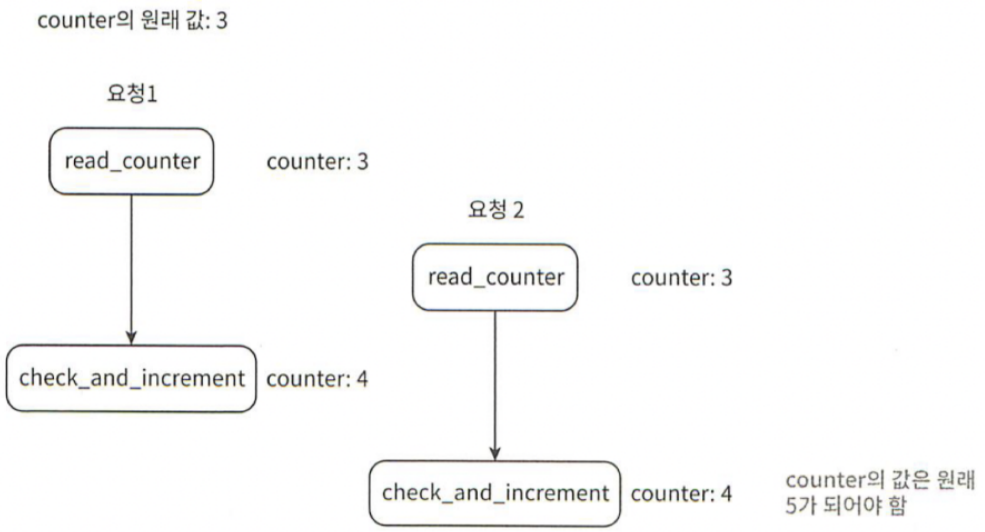
- RDB는 느리다.
- redis 와 같이 캐시 사용

처리율 한도 초과 트래픽 처리

- 요청 버림
- 메시지 큐에 보관 후 나중에 처리

분산 환경에서 처리율 제한 장치의 구현

- 해결해야하는 두 문제
    - race condition
        - 레디스에서 counter 값을 갱신하기 전 다른 요청’



        - Lua Script, Sorted Set으로 해결 / lock은 성능을 상당히 떨어트려 좋지 않다.
    - synchronization
        - 웹 계층이 무상태일 경우 서로 다른 제한 장치로 요청이 갈 수 있다.
        - sticky session 사용 가능하지만 확장이 까다롭고 유연하지 못한 방법
        - Redis와 같은 중앙 집중형 저장소를 사용

경성, 연성 처리율 제한

- Hard (경성) : 요청의 개수는 임계치를 절대 넘을 수 없다.
- Soft (연성) : 요청 개수는 잠시 동안은 임계치를 넘어설 수 있다.

다양한 계층에서의 처리율 제한

- application 계층에서의 처리를 알아봤다.
- ip 주소에 처리율 제한하는 방법도 있다. (Iptables)

처리율 제한을 회피하는 방법 (클라이언트)

- 클라언트측 캐시를 통해 api 호출 횟수 감소
- 짧은 시간동안 너무 많은 메시지를 보내지 않도록 한다.

- 예외적 상황으로 부터 gracefully한 복구될 수 있도록 예외, 에러 처리
- 재시도 로직은 충분한 back off 시간을 둔다.