5️⃣

5장. 안정 해시 설계

안정해시 사용 이유

수평적 규모 확장성을 달성하기 위해 **요청 또는 데이터를 서버에 균등하게** 나누는 것이 중요한데 이 목표를 달성하기 위해 안정해시를 보편적으로 사용합니다.

해시 키 재배치(rehash) 문제 (안정 해시가 해결하고자 하는 문제 정의)

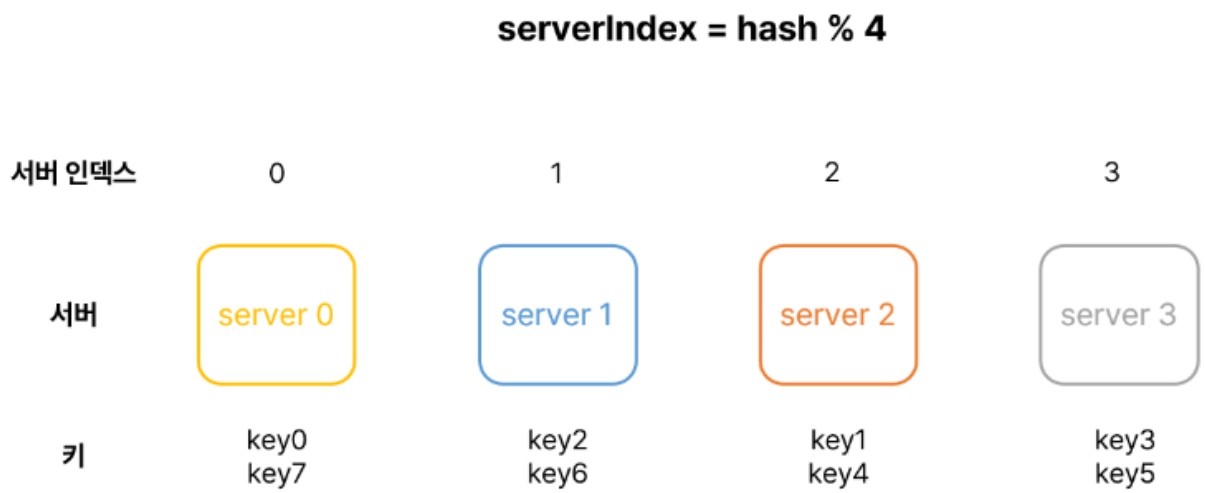
N개의 캐시 서버가 있고 부하를 균등하게 나누는 보편적인 방법은 아래와 같은 해시 함수를 사용하는 것

serverIndex = hash(key) % N (서버 수)

총 4대의 서버를 사용한다고 가정하고 각각의 키에 대한 해시 값과 서버 인덱스를 계산하면 다음과 같다.

키	해시	해시 % 4 (서버 인덱스)
key0	1034376036	0
key1	2365341234	2
key2	1579234085	1
key3	4912647603	3
key4	3459162480	2
key5	2016593051	3
key6	7895238741	1
key7	6749123912	0

이 후 키 값이 어떻게 분산되어 있는지 그림으로 나타내면 다음과 같다.



이 방법은 **서버 풀(server pool, 서버 갯수)의 크기가 고정**되어 있을 때, 그리고 **데이터 분포가 균등**할 경우 잘 동작합니다. 하지만 **서버가 추가**되거나, **기존 서버가 삭제**되면 **문제가 생깁**니다.

예를 들어 1번 서버가 장애가 발생해 동작을 중단 했다면, 서버 풀의 크기는 3으로 변하고, 결과적으로 해시 값은 변하지 않지만 나머지 연산을 적용하여 계산한 서버 인덱스 값은 달라질 것

키	해시	해시 % 3 (서버 인덱스)
key0	1034376036	0
key1	2365341234	0
key2	1579234085	2
key3	4912647603	0
key4	3459162480	0
key5	2016593051	2
key6	7895238741	0
key7	6749123912	2

serverIndex = hash % 3

기존 1번 서버에 있던 키 뿐만 아니라, 대부분의 키가 재분배되었습니다. 1번 서버가 죽으면 대부분의 캐시 클라이언트가 데이터가 없는 엉뚱한 서버에 접속하게 됩니다.

→ 그 결과 **대규모 캐시 미스(cache miss)**가 발생하게 될 것이고, 안정 해시는 이러한 문제를 효과적으로 해결하는 기술이다.

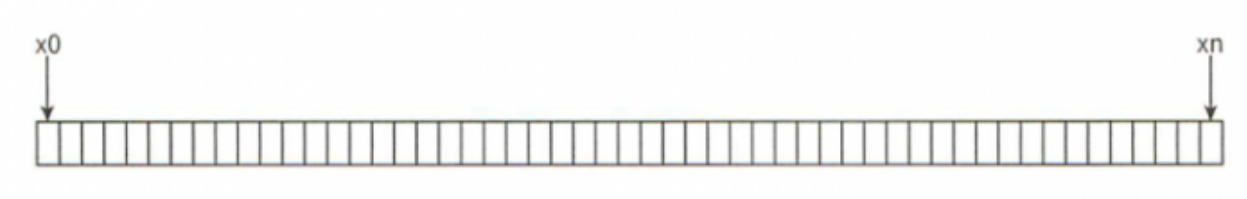
안정 해시

안정해시란?

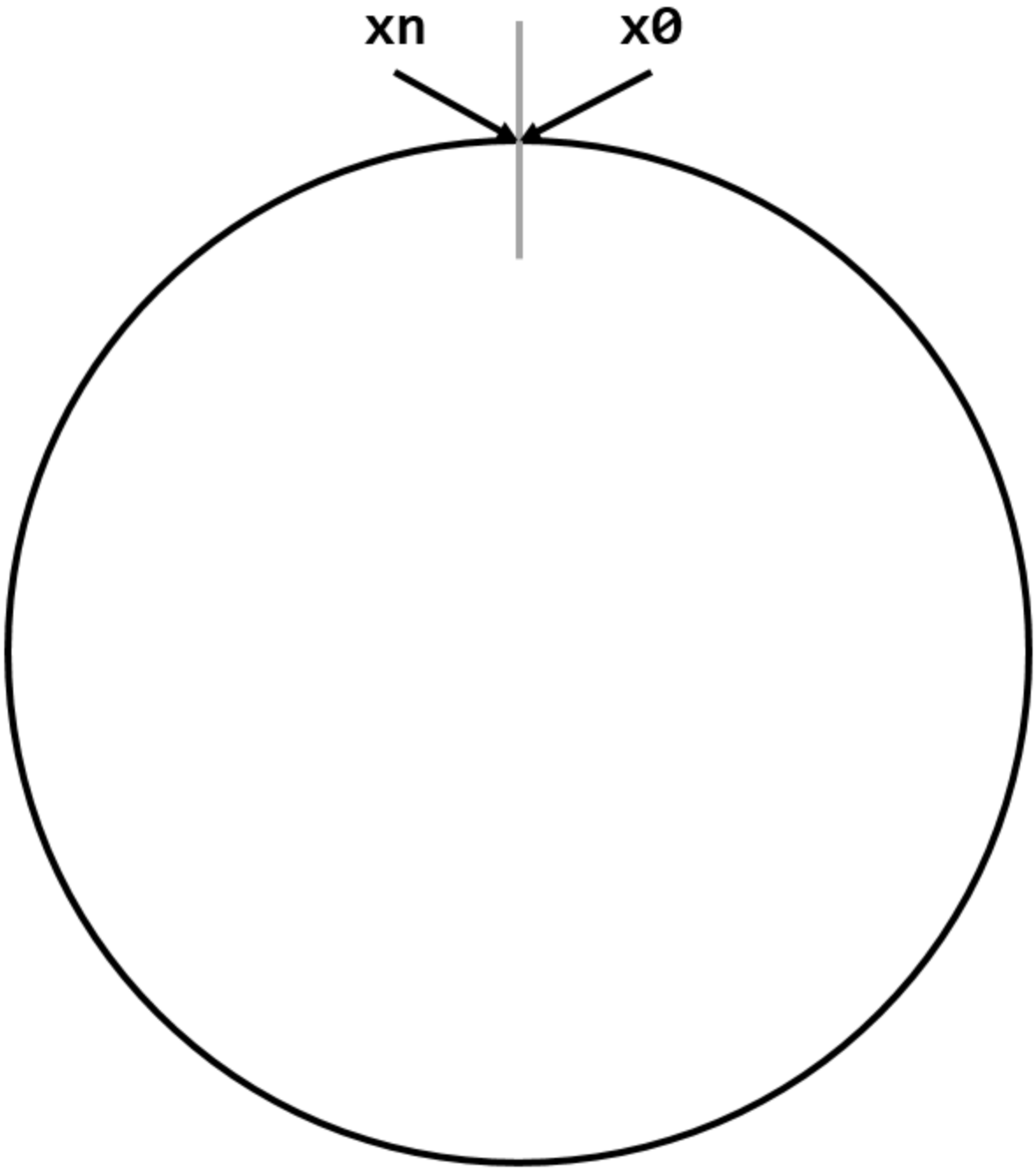
슬롯 수가 바뀌면 대부분의 키를 재배치하는 대부분의 전통적 해시 테이블과는 달리 안정 해시는 해시 테이블 크기가 조정될 때 **평균적으로 오직 k/n개의 키만 재배치** *(k = 키의 개수, n = 슬롯의 개수)* 하는 해시 기술입니다.

해시 공간과 해시 링

해시 함수 f는 SHA-1을 사용하고 그 함수의 출력범위는 x0, x1,…, xn과 같다고 하자. SHA-1의 해시 공간 범위는 0~2^160 -1 까지 알려져있고 x0 = 0, xn = 2^160-1로 나머지는 그 사이값을 갖게 된다.

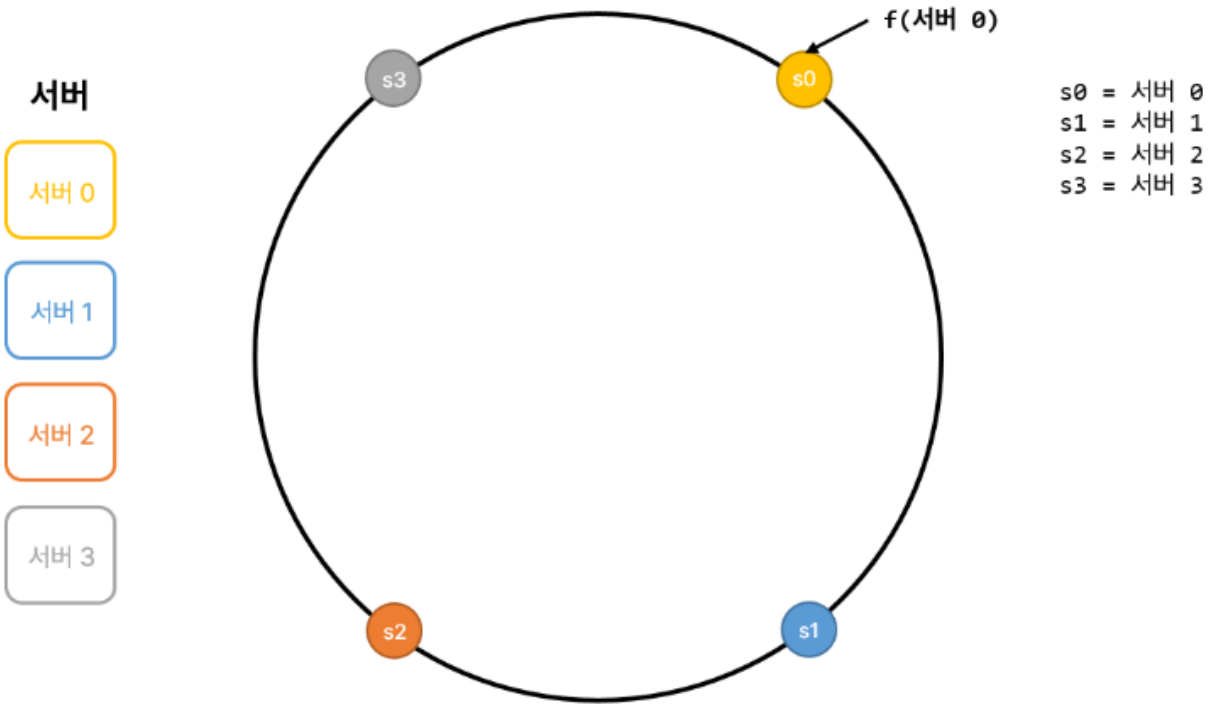


이러한 해시 공간 양쪽을 구부려 접으면 다음과 같은 **해시 링**이 만들어진다.



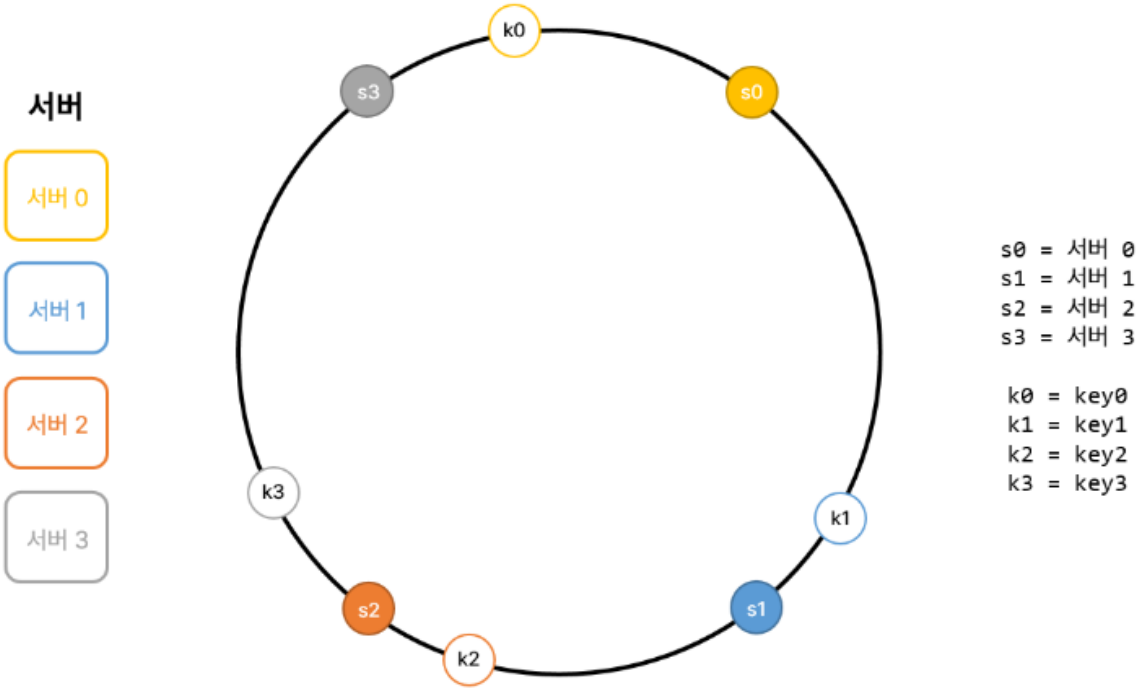
해시 서버

해시 함수 f를 사용하여 **서버 IP, 이름을 링 위 어떤 위치에 대응**시킬 것인지 알 수 있다.



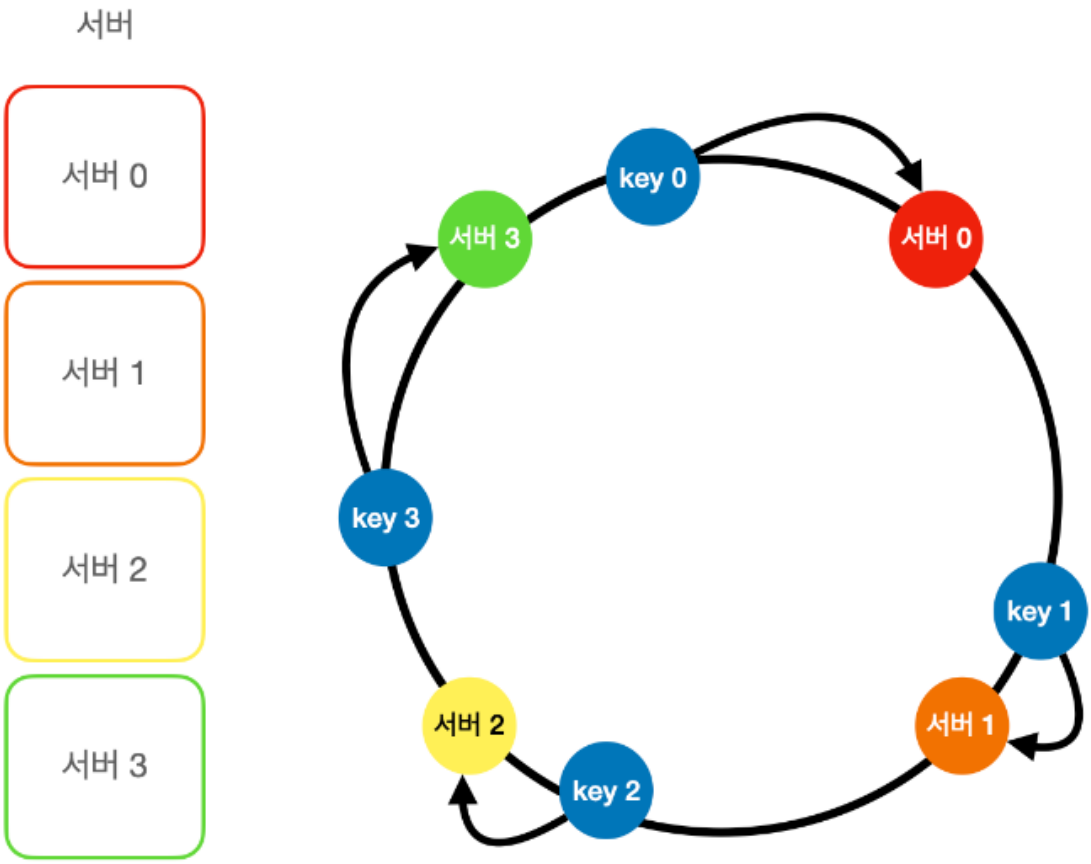
해시 키

현재 사용 중인 해시 함수는 나머지 연산을 사용하는 것과는 다르고 다음과 같이 키 값도 배치할 수 있다.



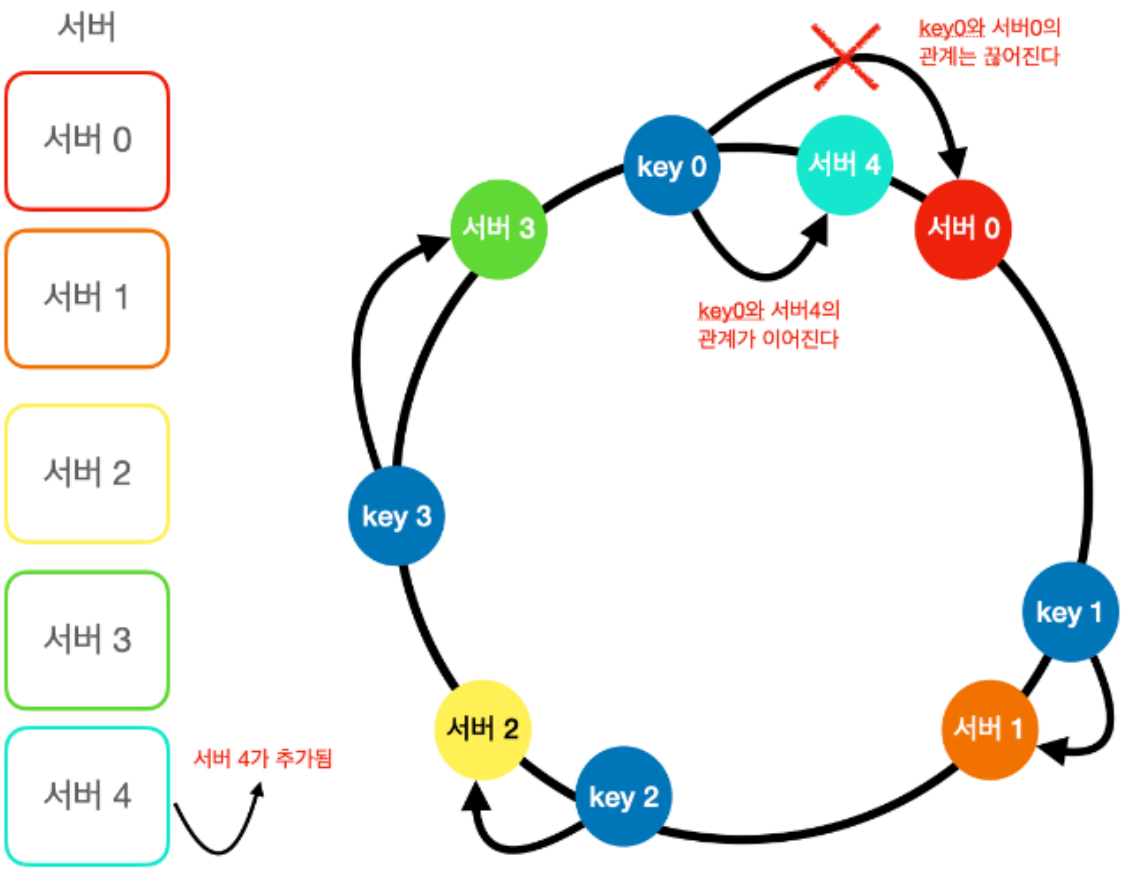
서버 조회

어떤 키가 저장되는 서버는, **해당 키의 위치로부터 시계 방향으로 링을 탐색해 나가면서 만나는 첫번째 서버**입니다.



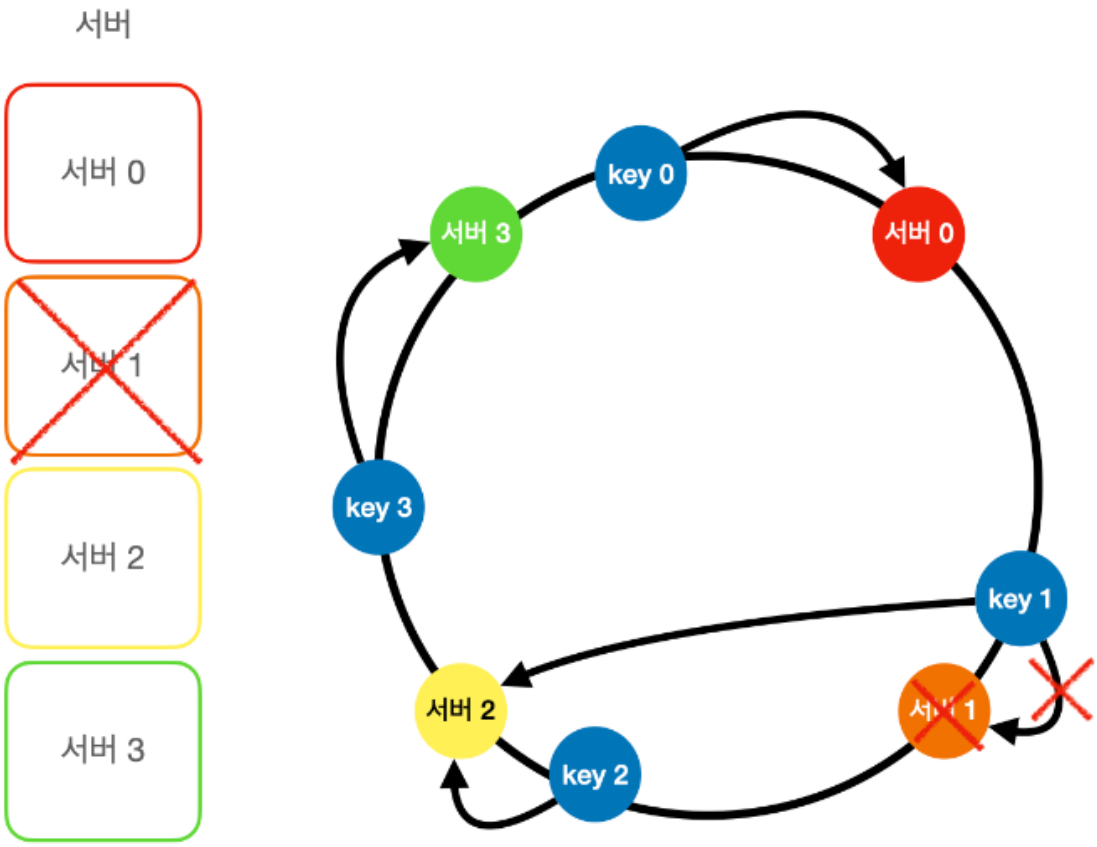
서버 추가

안정 해시를 사용했으므로, 서버를 추가하더라도 키 가운데 일부만 재배치하면 된다.



서버 삭제

하나의 서버가 제거 되면 키 가운데 일부만 재배치 된다.



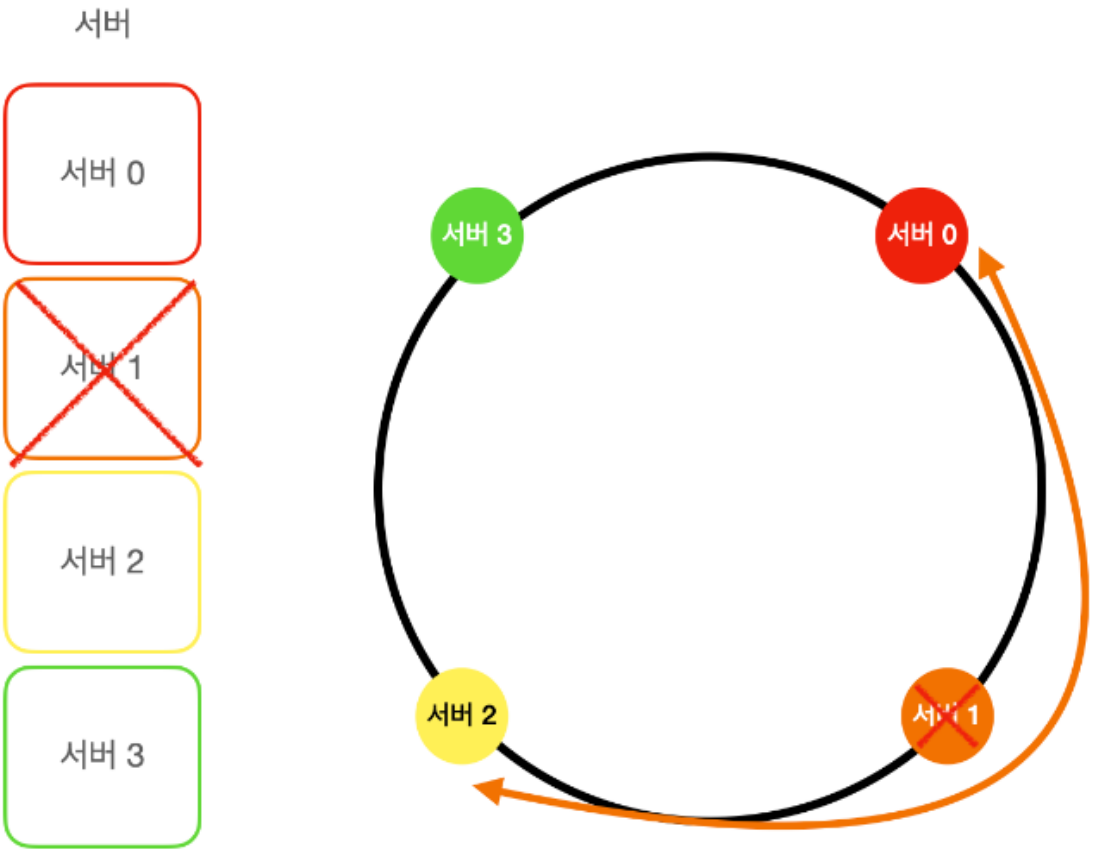
기본 구현법의 두 가지 문제

안성 해시의 기본 절차는 다음과 같다

- 서버와 키를 **균등 분포(uniform distribution) 해시 함수**를 사용해 햐사랑애 배치한다.
- 키의 위치에서 링을 시계 방향으로 탐색하다 만나는 최초의 서버가 키가 저장될 서버

위에는 두가지 문제가 존재한다.

1. 서버가 추가되거나 삭제되는 상황을 감안하면 **파티션(partion, 인접한 서버 사이의 해시 공간)의 크기를 균등하게 유지하는 것이 불가능** 하다.



실제 이런 경우 서버 2의 파이션이 다른 파티션에 비해 두 배로 커지는 것을 알 수 있다.

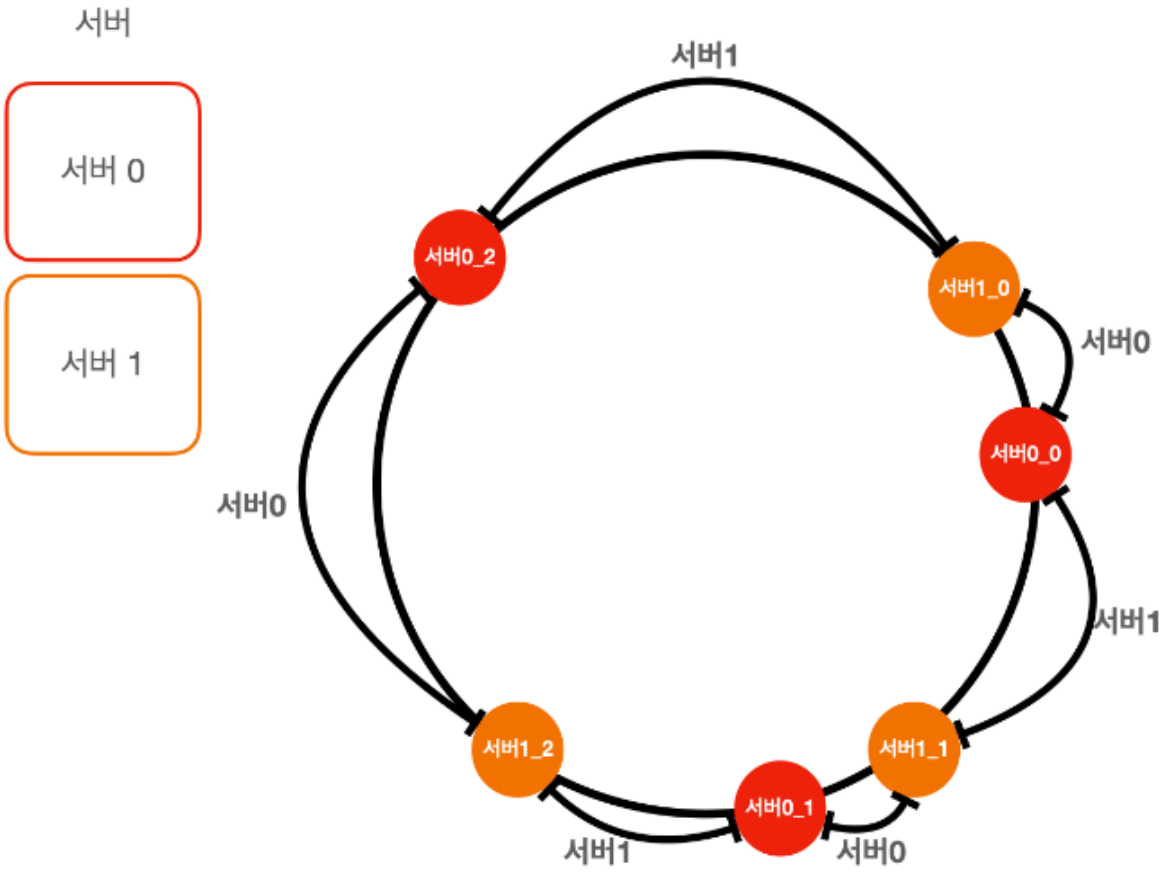
2. **키의 균등 분포 (uniform distribution)을 달성하기 어렵다.**

이 문제를 해결하기 위해 **가상 노드(virtual node) 또는 복제(replica)**라는 기법이 제안되었다.

가상 노드

가상 노드는 실제 노드 또는 서버를 가리키는 노드로서, 하나의 서버는 링 위에 여러 개의 가상 노드를 가질 수 있습니다.

즉, 각 서버는 하나가 아닌 여러 개 파티션을 관리해야 합니다.



가상 노드의 개수를 늘리면 **표준 편차가 작아져서 데이터가 고르게 분포**되므로 키의 분포는 점점 균등해집니다.

표준 편차는 데이터가 어떻게 퍼져 나갔는지를 보여주는 척도입니다.

가상 노드를 사용했을 경우 표준 편차 값은 평균의 5%(가상 노드 200개인 경우)에서 10%(가상 노드 100개인 경우) 사이로 가상 노드의 갯수가 늘어나면 표준 편차의 값이 떨어진다.

그러나 가상 노드 데이터를 저장할 공간이 많이 필요하게 될 것이다.

즉 타협적 결정이 필요한데, 시스템 요구 사항에 맞도록 가상 노드 개수를 적절히 조정해야 할 것이다.

재배치할 키 결정

- **서버 추가 시**

재배치에 영향을 받는 범위는 새로 추가된 노드부터 그 반시계 방향에 있는 첫 번째 서버까지

- **서버 삭제 시**

  재배치에 영향을 받는 범위는 삭제된 노드부터 그 반시계 방향에 있는 최초 서버까지

안정 해시의 장점

- 서버가 추가되거나 삭제될 때 재배치되는 키의 수 최소화
- 데이터가 균등하게 분포하게 되어 수평적 규모 확장성을 달성하기 쉽다.
- 핫스팟(트래픽 집중 지역) 키 문제를 줄인다. 특정 접근이 많은 샤드에 대한 접근이 지나치게 빈번하면 서버 과부하 문제가 생길 수 있다. → 안정 해시는 데이터를 좀 더 균등하게 분배해 이런 문제가 생길 가능성을 줄인다.