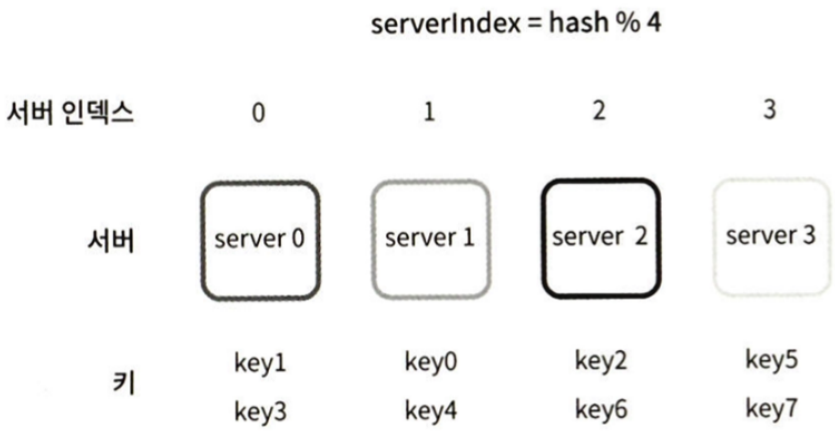
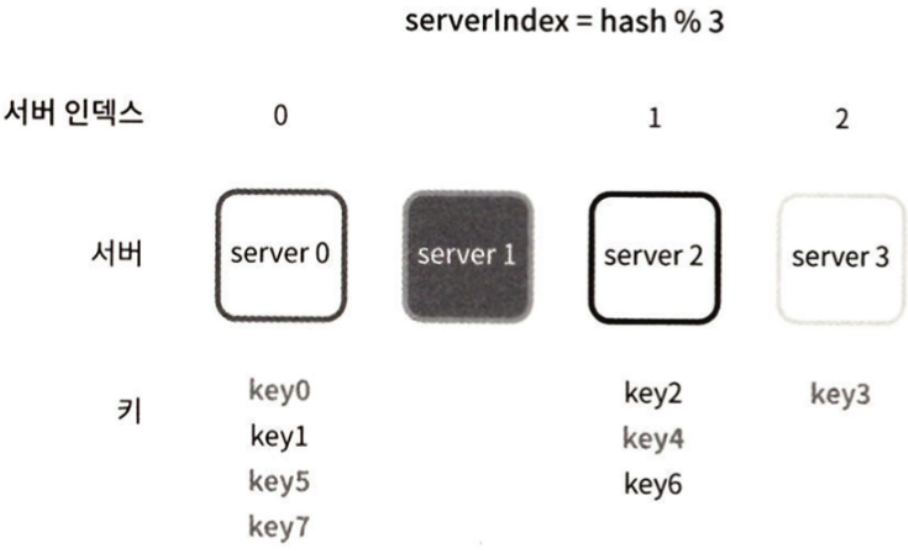
5장. 안정 해시 설계

수평적 규모 확장을 위해서는 요청 또는 데이터를 서버에 균등하게 나누는 것이 중요하다.
안정 해시는 이 목표를 달성하기 위해 보편적으로 사용하는 기술이다.

해시 키 재배치 문제

N개의 캐시 서버가 있다고 해보자. 이 서버들에 부하를 균등하게 나누는 보편적인 방법은 `hash_key % 서버 개수` 와 같은 해시 함수를 사용하는 것
하지만 이러한 해시 함수는 서버가 추가 되거나 삭제되는 경우 문제
→ 서버가 추가되거나 삭제되는 경우 해시 키의 재배치가 일어나는데, 이때 데이터의 불균형이 발생할 수 있기 때문
다음 예시는 4대의 서버 구성에서 서버 1대가 장애를 일으킨 경우의 해시 키 재배치의 문제점을 나타낸다.





- 장애가 발생한 1번 서버에 보관되어 있는 키 뿐만 아닌 대부분 키가 재분배되었다. 1번 서버가 죽으면 대부분 캐시 클라이언트가 데이터가 없는 엉뚱한 서버에 접속하게 된다는 뜻
  → **대규모 캐시 미스** 발생

안정 해시

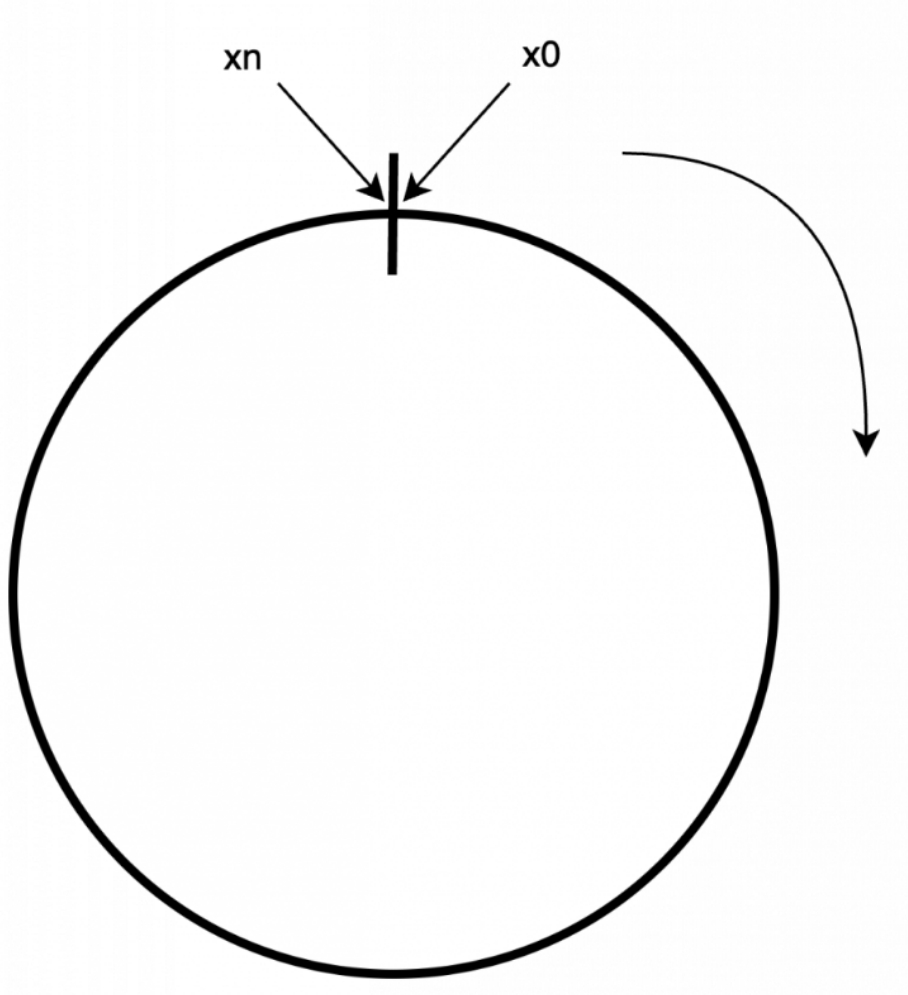
안정 해시 알고리즘은 MIT에서 처음 제안

- 서버와 키를 균등 분포(uniform distribution) 해시 함수를 사용해 해시 링에 배치한다.
- 키의 위치에서 링을 시계 방향으로 탐색하다 만나는 최초의 서버가 키가 저장될 서버이다.
  전통적인 해시 테이블은 슬롯의 수가 바뀌면 대부분 키를 재배치
  하지만, 안정 해시는 해시 테이블 크기가 조정될 때 평균적으로 **k(키의 개수)/n(슬롯의 개수)개의 키만 재배치**하는 해시 기술

해시 공간과 해시 링

해시 함수로 SHA-1을 사용한다고 한다.
해시 함수의 출력 값 범위를 x0부터 xn이라고 하자.
SHA-1의 해시 공간(hash space) 범위는 0부터 2^160-1까지로 알려져 있다.
따라서 x0은 0, xn은 2^160-1이며, 나머지 x1부터 xn -1 까지는 그 사이 값을 가짐

- 위의 해시 공간의 양쪽을 구부려 접으면 해시 링이 만들어 진다.



해시 서버

이 해시 함수를 사용하면 서버 IP나 이름을 이 링위의 어떤 위치에 대응시킬 수 있다.

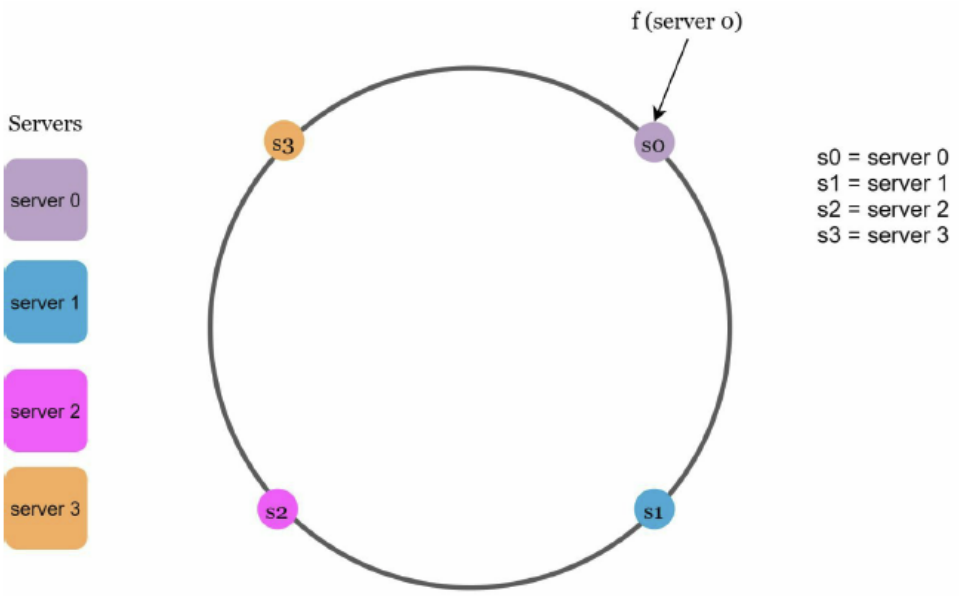


Figure 5-5

해시 키

여기 사용된 해시 함수는 "해시 키 재배치 문제"에 언급된 함수와는 다르며, 나머지 연산자는 사용하지 않고 있음에 유의한다. 캐시할 키 key0, key1, key3 또한 해시 링 위의 어느 지점에 배치할 수 있다.

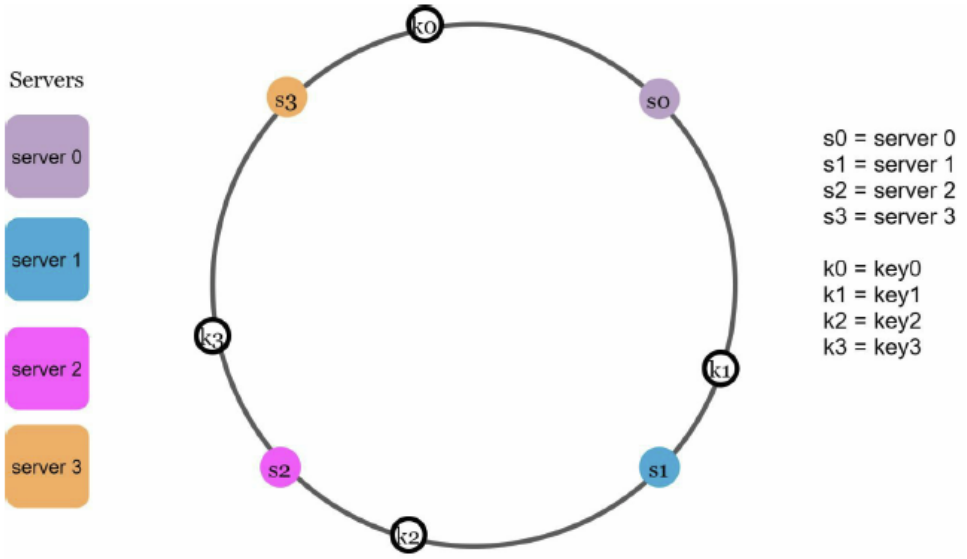


Figure 5-6

서버 조회

어떤 키가 저장되는 서버는, 해당 키의 위치로부터 **시계 방향으로 링을 탐색**해 나가면서 만나는 첫 번째 서버

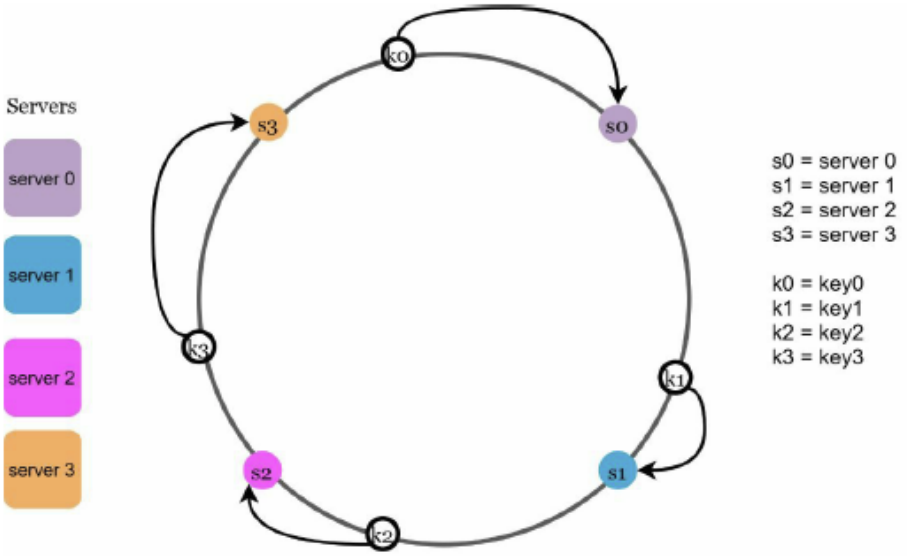
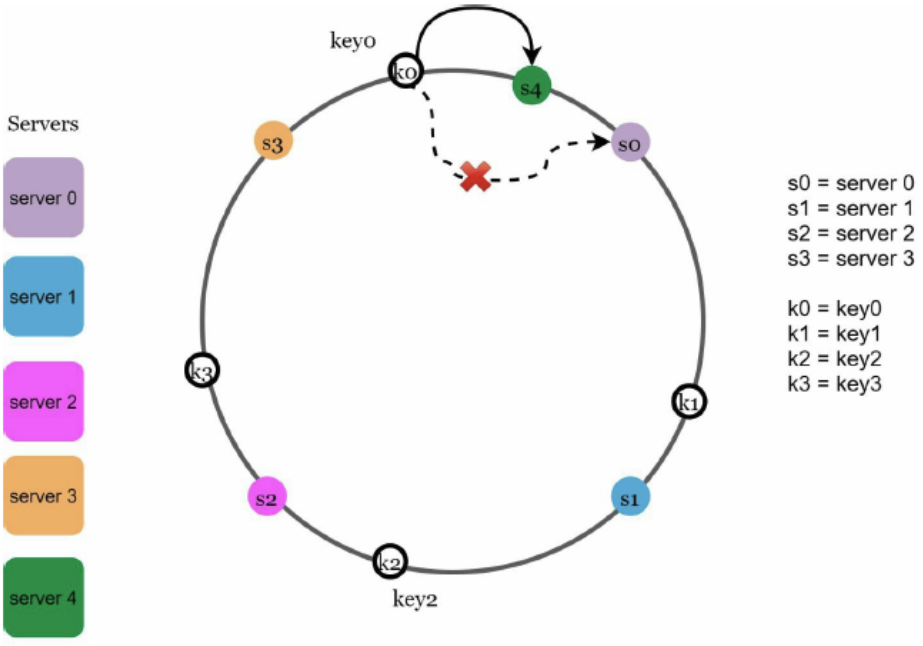


Figure 5-7

서버 추가

서버를 추가하더라고 키 가운데 **일부만 재배치**하면 된다.



- 새로운 서버 s4가 추가된 뒤에 key0만 재배치되는 것을 알 수 있다.

서버 제거

하나의 서버가 제거되면 키 가운데 **일부만 재배치**된다. 나머지 키에는 영향이 없다.

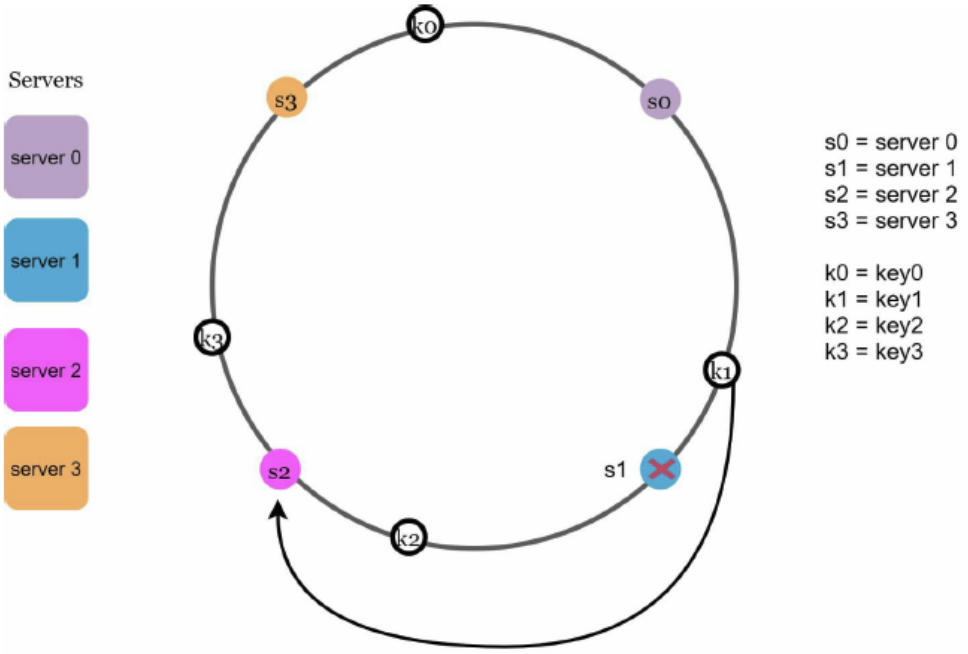


Figure 5-9

- s1이 삭제되었을 때 key1만이 s2로 재배치되는 것을 알 수 있다.

기본 구현법의 두 가지 문제

1. 서버가 추가되거나 삭제되는 사황을 감안하면 파티션의 크기를 균등하게 유지하는 게 불가능하다.

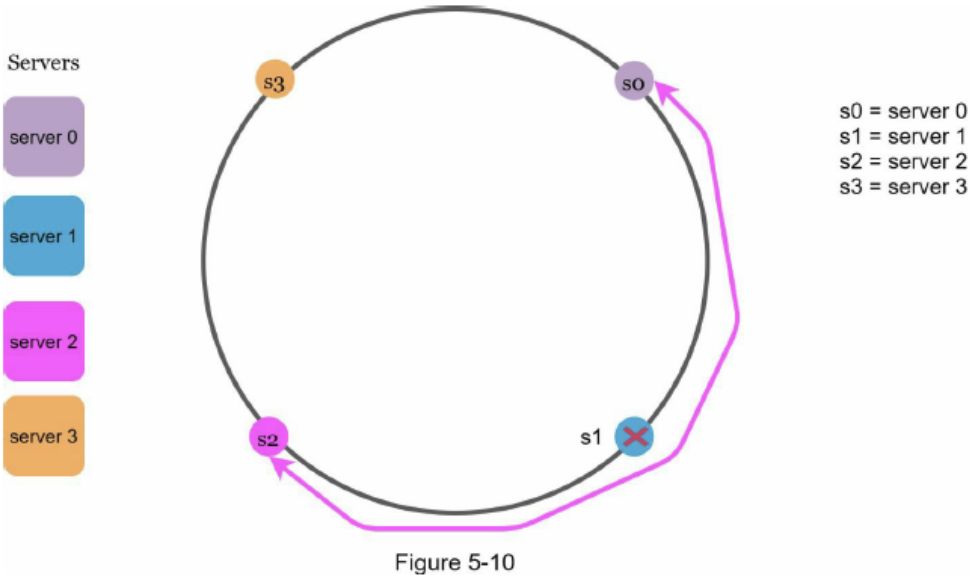


Figure 5-10

- s1이 삭제되는 바람에 s2의 파티션이 다른 파티션 대비 거의 두 배로 커지는 상황을 보여준다.

2. 키의 균등 분포를 달성하기가 어렵다.

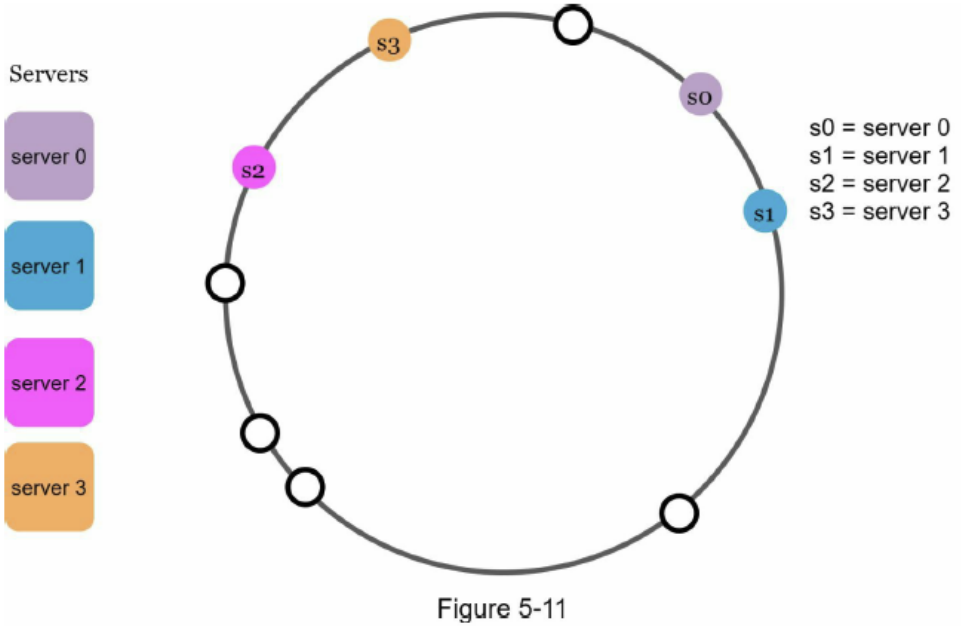
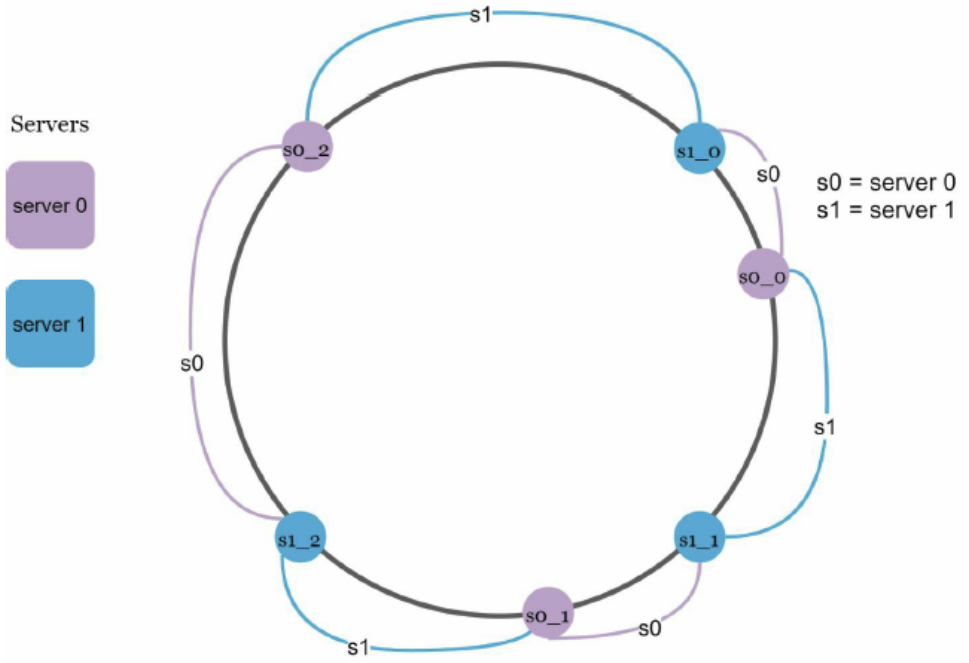


Figure 5-11

- s1, s3은 아무 데이터도 갖지 않는 반면, 대부분의 키는 s2에 보관되는 상황을 보여준다.

가상 노드

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



- s0, s1은 3개의 가상 노드를 갖는다. (여기서 숫자 3은 임의로 정한 것이며, 실제 시스템에서는 그보다 훨씬 큰 값이 사용)
- s0을 링에 배치하기 위해 s0하나만 쓰는 대신, s0_0, s0_1, s0_2의 세 개 가상 노드를 사용하였다. s1도 마친가지로 세 개의 가상 노드를 사용하였다.
  가상 노드의 개수를 늘리면 **표준 편차가 작아져서 데이터가 고르게 분포**된다. 하지만 가상 노드 데이터를 저장할 공간은 더 많이 필요하게 된다. → **타협적 결정** 필요

안정 해시 이점

- 서버가 추가되거나 삭제될 때 재배치되는 키의 수가 최소화 된다.
- 데이터가 보다 균등하게 분포하게 되므로 수평적 규모 확장성을 달성하기 쉽다.
- 핫스팟(hotspot) 키 문제를 줄인다. 특정한 샤드(shard)에 대한 접근이 지나치게 빈번하면 서버 과부하 문제가 생길 수 있다. 안정 해시는 데이터를 좀 더 균등하게 분배하므로 이런 문제가 생길 가능성을 줄인다.

안정 해시 사용 사례

- 아마존 다이나모 데이터베이스(DynamoDB)의 파티셔닝 관련 컴포넌트
- 아파치 카산드라(Apache Cassandra) 클러스터에서의 데이터 파티셔닝
- 디스코드(Discord)채팅 어플리케이션
- 아카마이(Akamai) CDN
- 매그래프(Meglev) 네트워크 부하 분산기