안정해시 설계

왜 안정해시 설계를 알아야 할까?

안정해시설계의 이론은 간단하다. 컴퓨터 구조에도 맞닿는 부분이 있다.

일단 안정해시설계가 나온 배경부터 이해해보자.

해시는 요청 또는 데이터를 서버에 균등하게 나누는 것이 중요하다.

serverIndex = hash(key) % N(서버의 개수)

여기서 hash() 는 어떤 알고리즘에 의한 함수.

안정해시설계가 나온 이유?

문제점

해시 키 재배치(rehash) 문제

모듈러 연산을 통해 부하를 균등하게 나눌 수 있는 서버 4대가 있다.

111,222,333,444,555,666,777,888,999 의 해시를 갖는 키가 있다고 가정해보자.

키	해시	해시 % 4(서버 인덱스)
key 0	111	1
key 1	222	2
key 2	333	3
key 3	444	0
key 4	555	1
key 5	666	2
key 6	777	3
key 7	888	0
key 8	999	1

서버	해시키
Server 1	key0, key4, key7
Server 2	key 1, key 5
Server 3	key 2, key 6
Server 4	key 3, key 7

균등하게 배포될 수 있다. 그래서 예시에서는 해시의 키가 이미 정렬된 상태였기 때문에 가능했던 것이며, 여기서 만약 Server 1이 죽으면 어떻게 될까? 다시 재배치를 해야될텐데.

모듈러 계산

해시 % 4(서버 인덱스) 에서

해시 % 3(서버 인덱스) 으로 변하게 된다.

그럼 Server 1 에서 가졌던 key0, key4, key7 키뿐만 아니라 모든 해시키가 재배치된다.

그 결과로 기존에 Server1,2,3,4 를 쓰던 클라이언트는 엉뚱한 서버에서 값을 찾을 테고 그 결과 대뮤고 캐시미스(cache miss)가 발생한다.

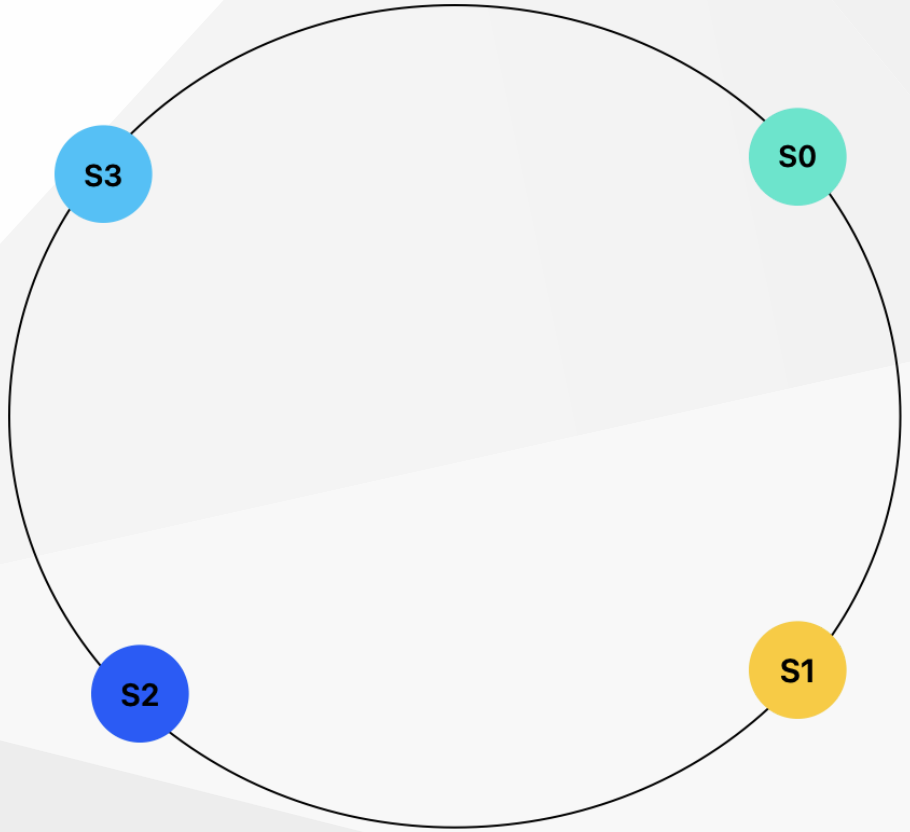
안정 해시

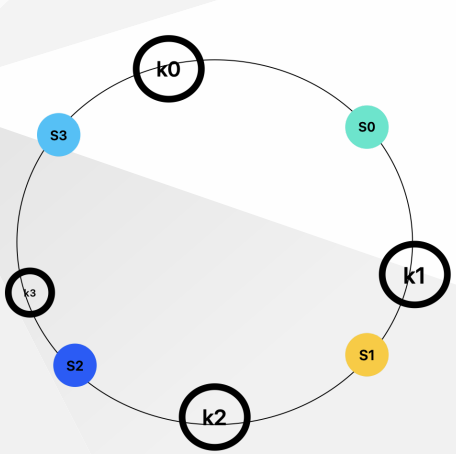
그래서 나온게 안정 해시

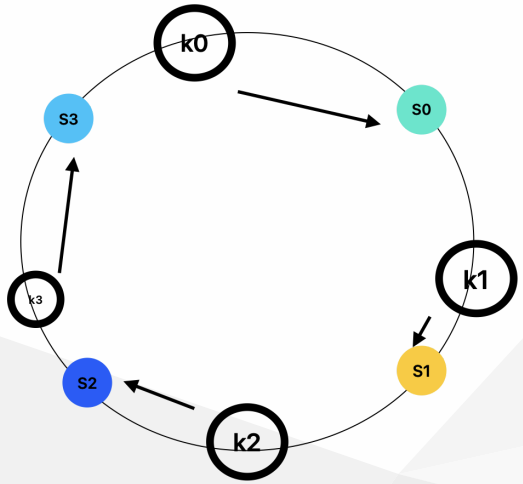
안정해시는 해시 테이블이 조정될 때 **평균적으로 오직 k/n개** 의 키만 재배치하는 해시 기술.

여기서 해시 공간과 해시 링 등장인물이 등장

S1,2,3 는 해시 서버







해시 키는 k0,k1,k2,k3 로 표현된다. 이때 각 키는 모듈러 연산으로 나온 결과물을 의미하지 않는다.

이때 각각의 키는 시계방향으로 링을 탐색하면 마주치는 서버에 저장.

만약 s0 이 사라진다면?

또는 s0_1이 추가된다면

다시 말하지만 안정해시는 평균적으로 오직 k/n개 키만 재배치하는 해시 기술이다.

위 방식의 안정해시 기술은 2가지 문제점

첫번째
서버가 추가되거나 삭제되는 상황을 감안하면 파티션(partition)의 크기를 균등하게 유지하는 게 불가능하다는 것(파티션은 인접한 서버 사이의 해시 공간)

어떤 서버는 굉장히 작은 해시 공간을 할당 받고, 어떤 서버는 굉장히 큰 해시 공간을 할당 받는 상황이 가능하다는 것.

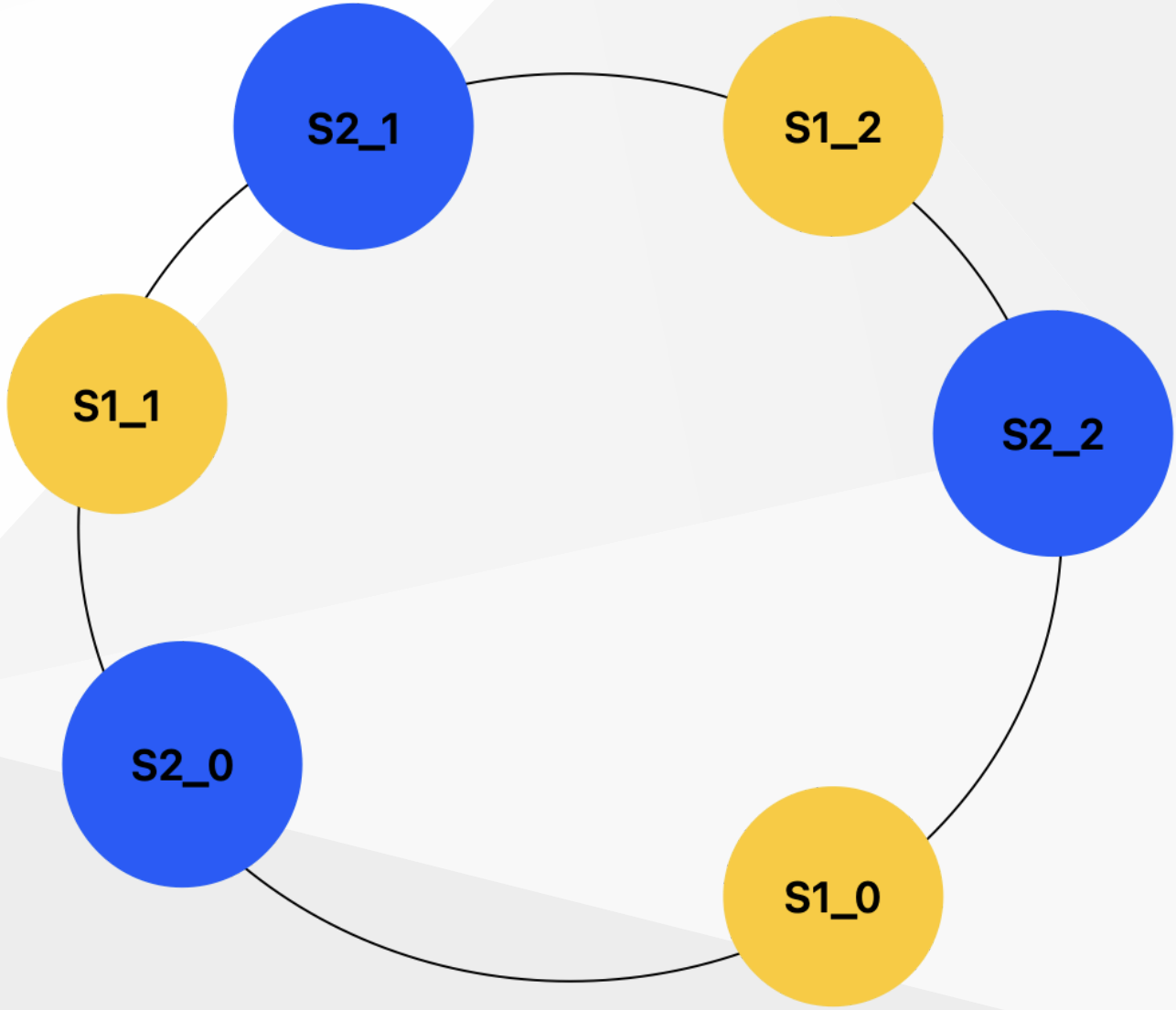
두번째
키의 균등 분포(unifrom distribution)를 달성하기가 어렵다는 것. 균등한 키의 분포를 해결하기 위해서 제안된 기법은 가상 노드(virtual node) 또는 복제(replica)라 불리는 기법

어떻게 해결할 수 있을까? 앞에서 언급한 **가상 노드** 를 활용

가상 노드의 개수를 늘리면 키의 분포는 점점 더 균등.

표준 편차(standard deviation)가 작아져서 데이터가 고르게 분포되기 때문.

타협적 결정(tradeoff)



코드 살펴보기