5장. 안정 해시 설계

Created	@October 6, 2022 2:47 PM
Progress	In Progress

📃 **학습 TODO list**

- [ ] SHA-1

> *수평적 규모 확장성을 달성하기 위해서는 요청 또는 데이터를 서버에 균등하게 나누는 것이 중요하다. → 보편적으로 **안정 해시**를 사용한다.*

👩‍💻 `안정 해시`의 이점

- 서버가 추가되거나 삭제될 때 재배치되는 키의 수가 최소화된다.
- 데이터가 보다 균등하게 분포하게 되므로 `수평적 규모 확장성`을 달성하기 쉽다.
- `핫스팟(hotspot) 키 문제`를 줄인다. 특정한 샤드(shard)에 대한 접근이 지나치게 빈번하면 서버 과부하 문제가 발생할 수 있다.

👩‍💻 `안정 해시`가 사용되는 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

5.1. 해시 키 재배치(rehash) 문제

> *해시 기술이 풀려고 하는 문제*

- N개의 캐시 서버가 있다고 할 때, 서버들에 부하를 균등하게 나누는 보편적인 방법은 아래의 해시 함수를 사용하는 것이다.

  `serverIndex = hash(key) % N`

  - 이 방법은 `서버 풀(server pool)`의 크기가 고정되어 있을 때, 그리고 `데이터 분포`가 균등할 때는 잘 동작한다.
  - 하지만 서버가 삭제되면 문제가 생긴다. → 장애가 발생한 서버에 보관되어 있는 키 뿐만 아니라 대부분의 키가 재분배되어, 캐시 클라이언트가 데이터가 없는 엉뚱한 서버에 접속하여 대규모 `캐시 미스(cache miss)`가 발생할 것

이다.

- 이 문제는 `안정 해시`로 효과적으로 해결할 수 있다.

5.2. 안정 해시

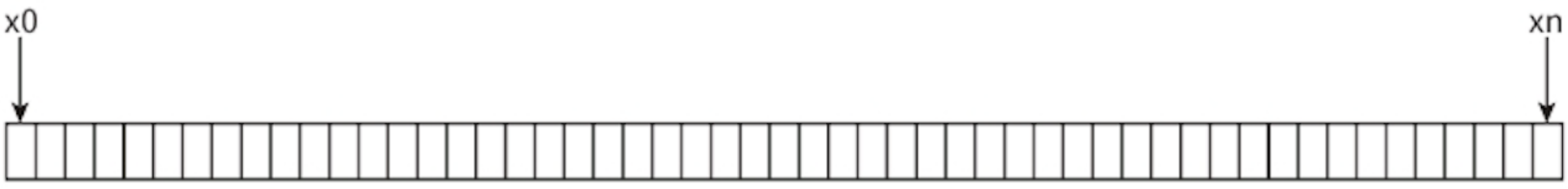
> *안정 해시(consistent hash)는 해시 테이블 크기가 조정 될 때 평균적으로 오직 k/n개의 키만 재배치하는 해시 기술이다.(k: 키의 개수, n: 슬롯(slot)의 개수)*

> *전통적 해시 테이블은 슬롯의 수가 바뀌면 거의 대부분 키를 재배치한다.*

5.2.1. 해시 공간과 해시 링

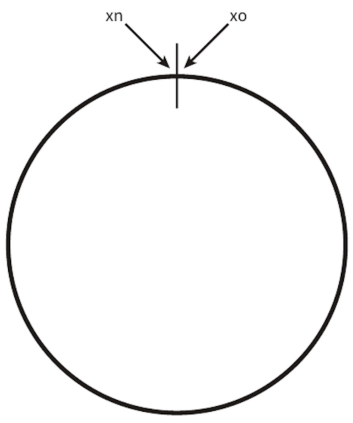
> *안정 해시의 동작 원리*

- 해시 함수 f: SHA-1
  - 출력 값 범위: x0, x1, x2, x3, …, xn
  - 해시 공간(hash space) 범위: 0 ~ (2^{160} - 1) → x0 = 0, xn = 2^{160} - 1



해시 공간

  - 해시 링(hash ring): 해시 공간의 양쪽을 구부려 접으면 만들 수 있다.



해시 링

5.2.2. 해시 서버

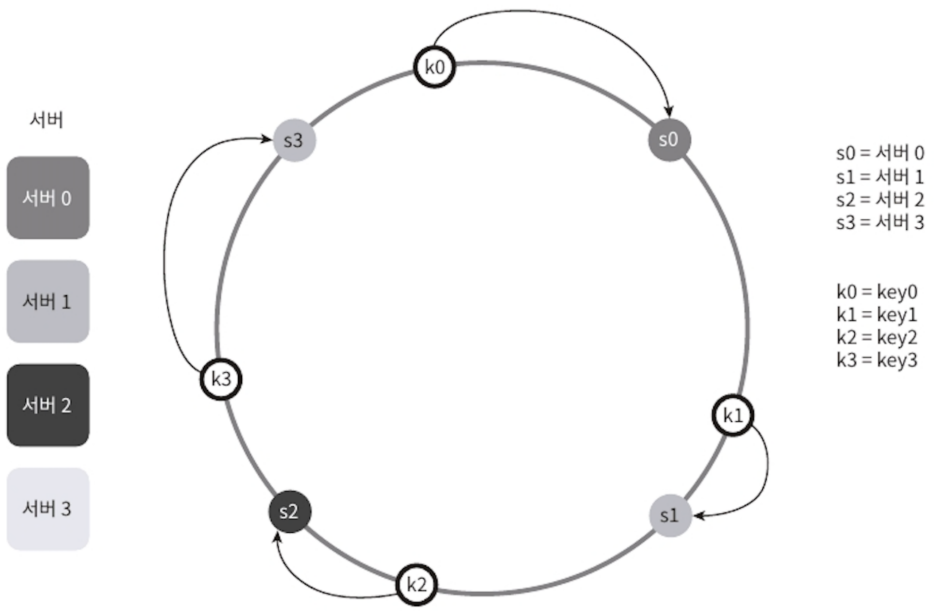
- 서버 IP나 이름을 해시 링 위에 대응시킬 수 있다.

5.2.3. 해시 키

- 캐시할 키를 해시 링 위의 어느 지점에 배치할 수 있다.

5.2.4. 서버 조회

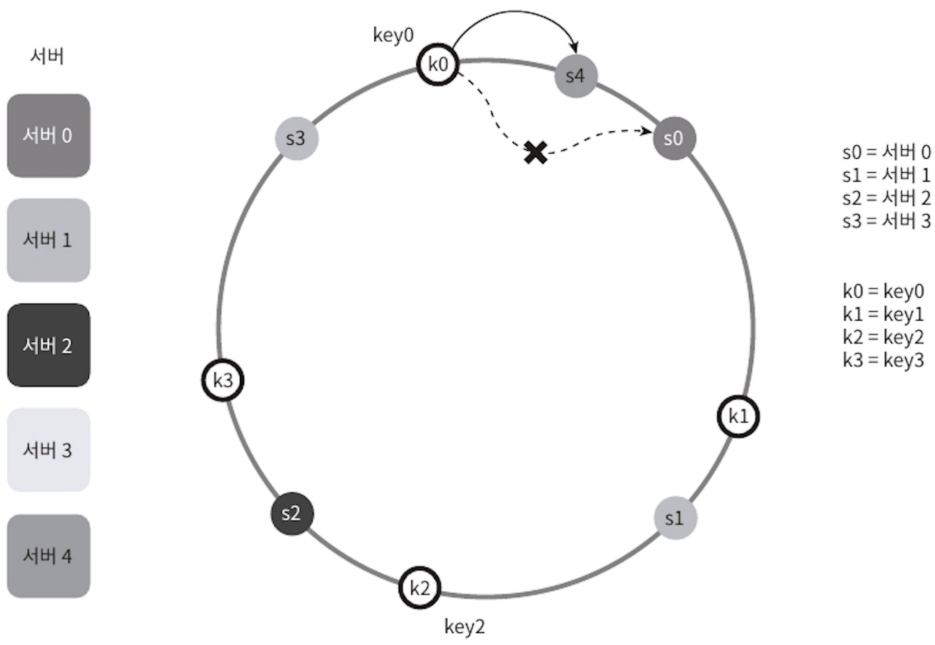
> *서버와 키를 해시 링 위에 배치해보자.*



- 어떤 키가 저장되는 서버는 해당 키의 위치로부터 시계 방향으로 링을 탐색해 나가면서 만나는 첫 번째 서버이다.
  - `key0`는 `서버0`에 저장된다.
  - `key1`는 `서버1`에 저장된다.
  - `key2`는 `서버2`에 저장된다.
  - `key3`는 `서버3`에 저장된다.

5.2.5. 서버 추가

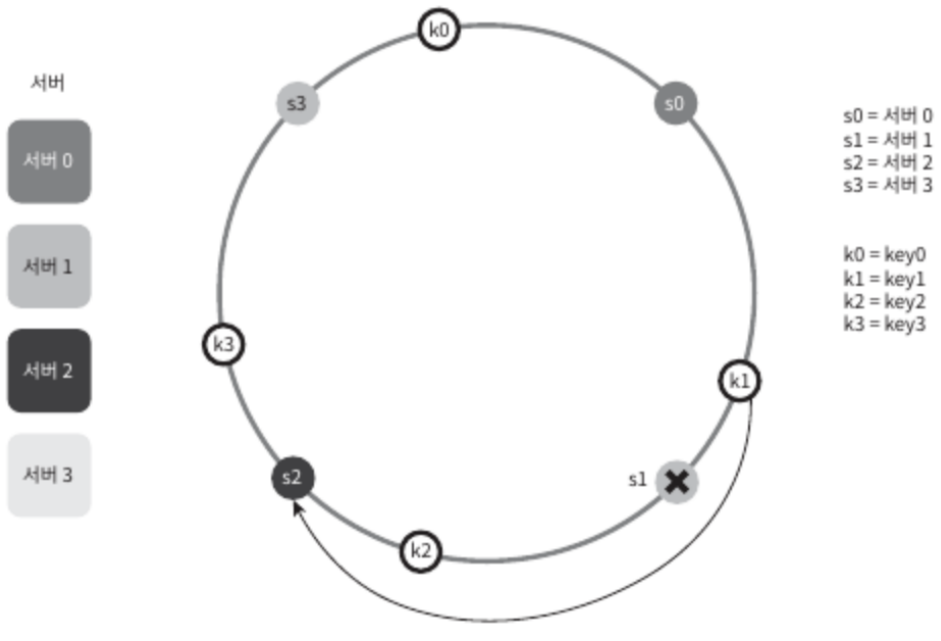
> *서버를 추가하더라도 키 가운데 일부만 재배치하면 된다.*



- `서버4`가 추가된 뒤에 `key0`만 재배치 되었고, `key1`, `key2`, `key3`은 같은 서버에 남는다.
  - `key0`에서 시계 방향으로 순회했을 때 처음으로 만나게 되는 서버가 `서버4`이다.
  - 다른 키들은 재배치되지 않는다.

5.2.6. 서버 제거

> *하나의 서버가 제거되면 키 가운데 일부만 재배치된다.*



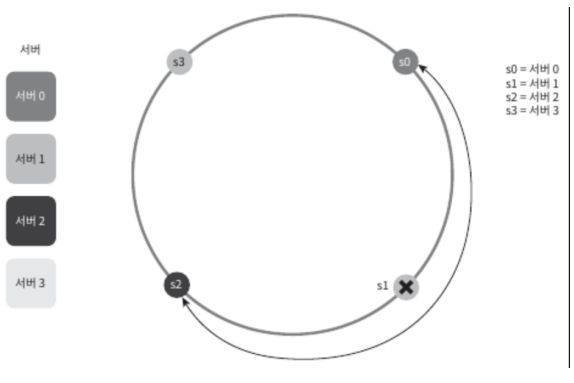
- `서버1`이 삭제 되었을 때 `key1`이 `서버2`로 재배치되었다.
  - 다른 키들은 재배치되지 않는다.

5.2.7. 기본 구현법의 두 가지 문제

- `안정 해시 알고리즘`: MIT에서 처음 제안되었다.
  - 서버와 키를 `균등 분포(uniform distribution) 해시 함수`를 사용해 해시 링에 배치한다.
  - 키와 위치에서 링을 시계 방향으로 탐색하다 만나는 최초의 서버가 키가 저장될 서버이다.

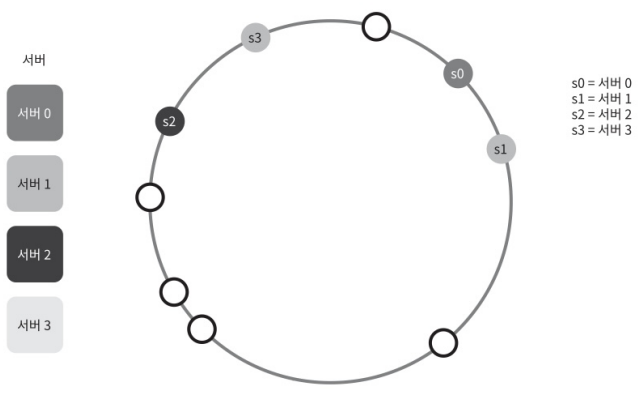
> *안정 해시 알고리즘의 문제점*

1. 서버가 추가되거나 삭제되는 상황을 감안하면 `파티션(partition)`의 크기를 균등하게 유지하는 게 불가능하다.
   - `파티션`: 인접한 서버 사이의 해시 공간
   - 어떤 서버는 굉장히 작은 해시 공간을 할당 받고, 어떤 서버는 굉장히 큰 해시 공간을 할당 받을 수 있다.



`서버1`이 삭제되어 `서버2`의 파티션이 다른 파티션 대비 거의 두 배로 커지는 상황

2. 키의 `균등 분포(uniform distribution)`를 달성하기가 어렵다.

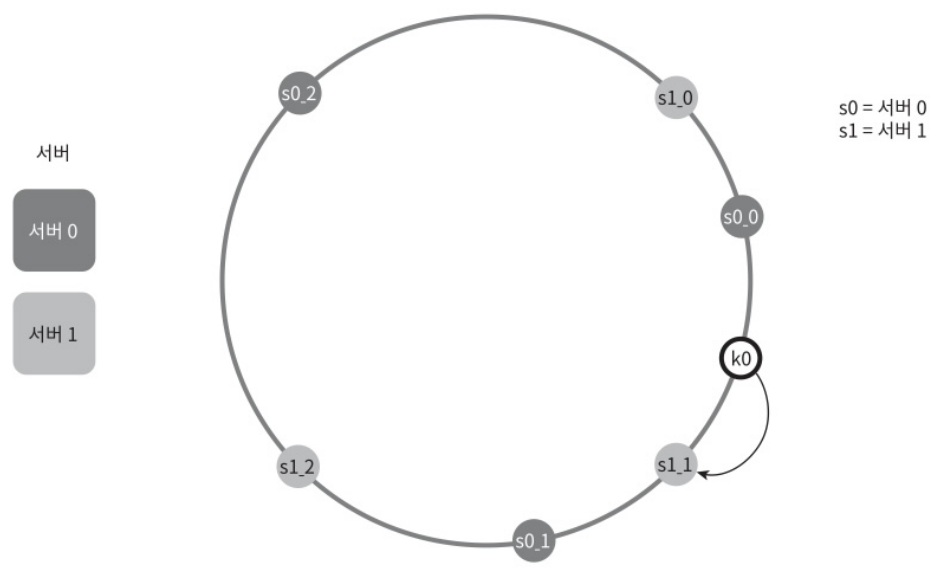


`서버1`과 `서버3`은 아무 데이터도 갖지 않는 반면, 대부분의 키가 `서버2`에 보관되는 상황

5.2.8. 가상 노드

> *가상 노드(virtual node), 복제(replica) 방법으로 안정 해시 알고리즘의 문제점을 해결할 수 있다.*

- `가상 노드(virtual node)`: 실제 노드 또는 서버를 가리키는 노드로서, 하나의 서버는 링 위에 여러 개의 가상 노드를 가질 수 있다.



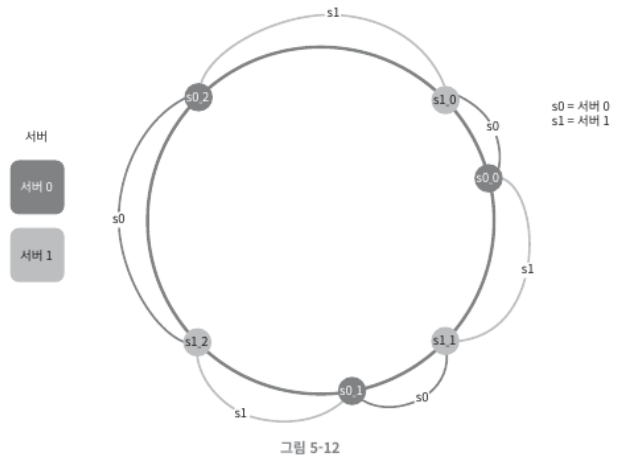


그림 5-12

- 키의 위치로부터 시계방향으로 링을 탐색하다 만나는 최초의 가상 노드가 해당 키가 저장될 서버가 된다.

- `가상 노드`의 개수를 늘리면 키의 분포는 점점 더 균등해진다.
  - `표준 편차(standard deviation)`이 작아져서 데이터가 고르게 분포된다.
    - `표준 편차`: 데이터가 어떻게 퍼져 나갔는지를 보이는 척도
    - 100~200개의 가상 노드를 사용했을 경우 표준 편차 값은 평균의 5%(가상 노드 200개)에서 10%(가상 노드 100개) 사이다. → 가상 노드 개수를 늘리면 `표준편차` 값은 떨어진다. 하지만 가상 노드 데이터를 저장할 공간을 더 많이 필요해진다.(`tradeoff`)

5.2.9. 재배치할 키 결정 → 5.2.5. 서버 추가, 5.2.6. 서버 제거