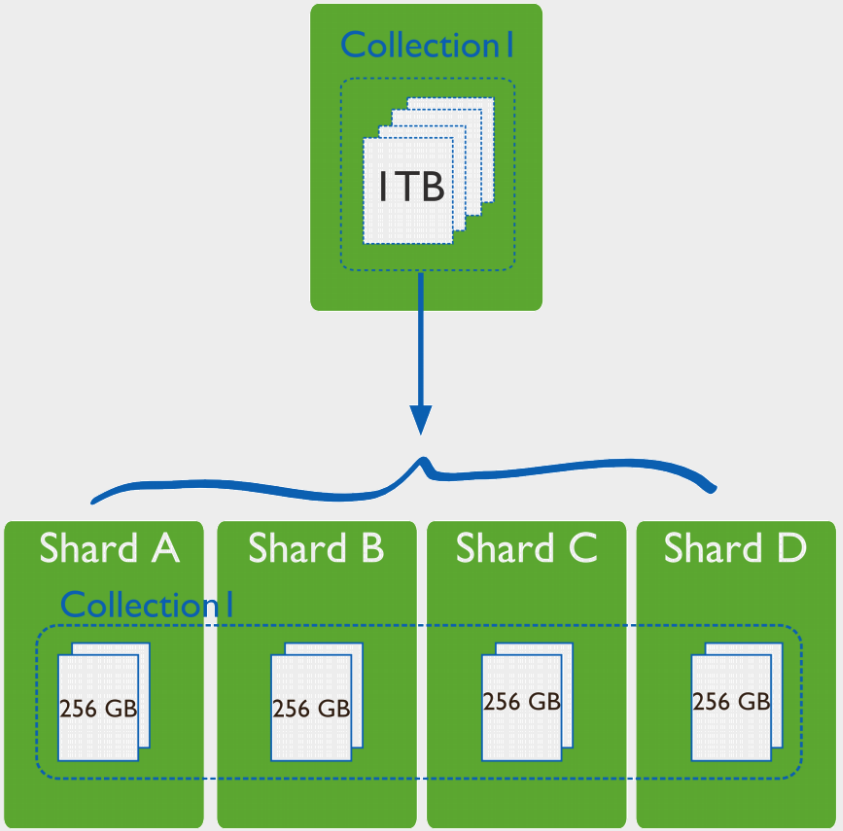
대규모 데이터 관리를 위한 mongoDB sharding 전략

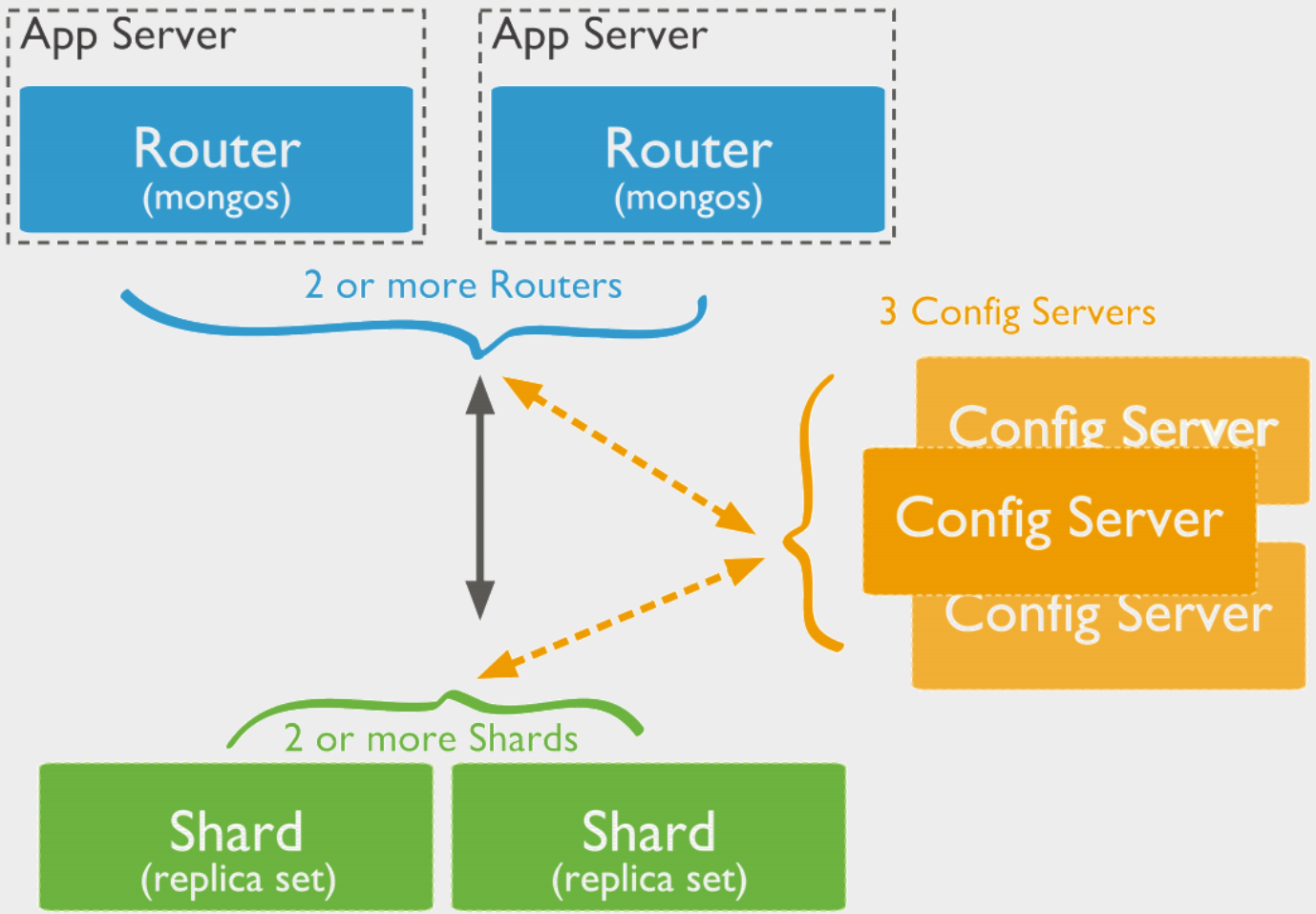
AGENDA

1. mongoDB sharding 개요
2. mongoDB auto sharding 문제점
3. 전략 방안
4. 사례 연구

1. mongoDB sharding 개요 – sharding 이란?



1. mongoDB sharding 개요 – sharding in mongoDB



1. mongoDB sharding 개요 – data partitioning

● Shard keys

- ✓ Shard key 는 하나의 collection 을 shard 하기 위해 선택한 key
- ✓ Shard key 는 indexed field 또는 복합 field
- ✓ MongoDB 는 shard key 값들을 chunk 들로 나누고 각 샤드에 chunk 들을 균등하게 분산하여 저장

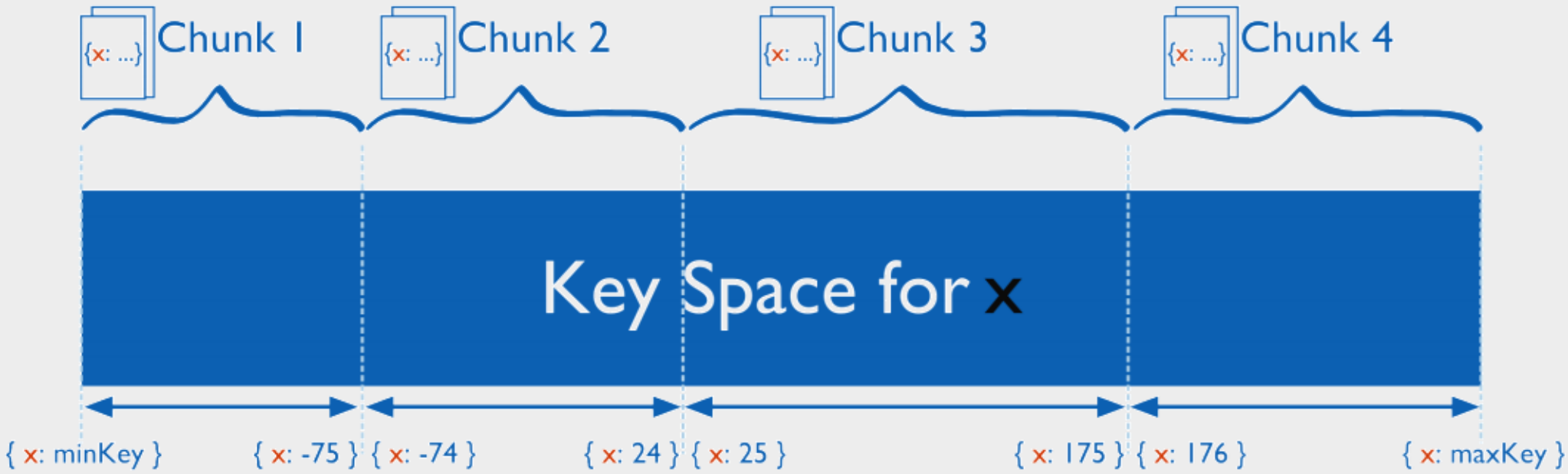
● Range Based Sharding

- ✓ Range based sharding 은 shard key 값에 의해 shard 를 결정하는 범위로 data set 을 나누는 방식

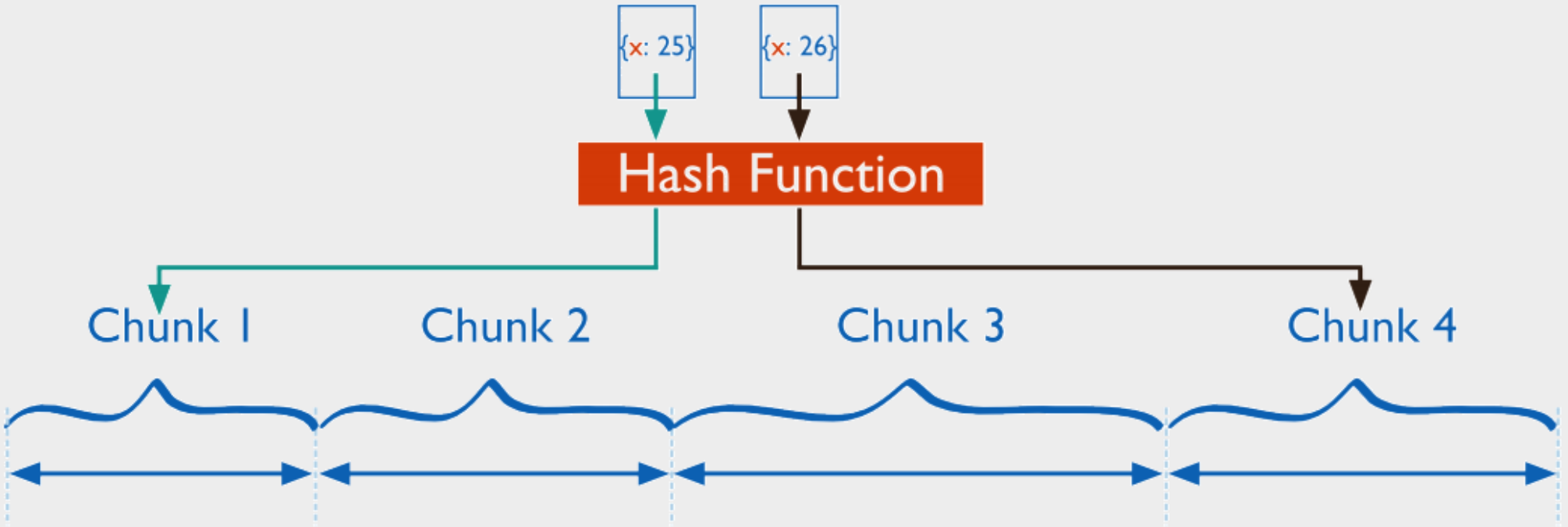
● Hash Based Sharding

- ✓ Hash based sharding 은 field 값의 hash 를 계산 하고, chunks 를 만들기 위해 해당 hash 를 사용

1. mongoDB sharding 개요 – Range Based Sharding



1. mongoDB sharding 개요 – Hash Based Sharding



1. mongoDB sharding 개요 – Range vs. Hash Based Sharding

● Range Based Sharding

- ✓ 효율적인 범위 검색. Query router 가 쉽게 어떤 chunk 가 해당 범위에 속하는지 결정
  해당 chunk 가 저장된 shard 에만 query 를 요청함
- ✓ 단점으로 불균등한 데이터 분산이 되면, 그 경우 모든 요청이 한 쪽으로 쏠려
  부하 분산 또는 확장이 잘 되지 않을 수 있다.

● Hash Based Sharding

- ✓ Range Based Sharding 의 단점인 불균등 데이터 분산을 해결하여 균등 데이터 분배가 가능
- ✓ 단점으로 Hashed key 는 chunk 와 shard 들에 랜덤한 분산 저장으로 range query 와 달리 소수의 shard 에만 query 를 요청하기 보다는 대부분의 모든 shard 에 query 를 발생시켜야 결과를 얻을 수 있다.

1. mongoDB sharding 개요 – Balanced Data Distribution

● Unbalanced Data Distribution

- ✓ 새로운 데이터 또는 새로운 서버의 추가로 인해 cluster 내부 데이터 불균형이 발생.
- ✓ 특정 shard 가 다른 shard 보다 더 많은 chunk 를 담거나 다른 chunk 에 비해 훨씬 큰 size 를 가지게 됨.

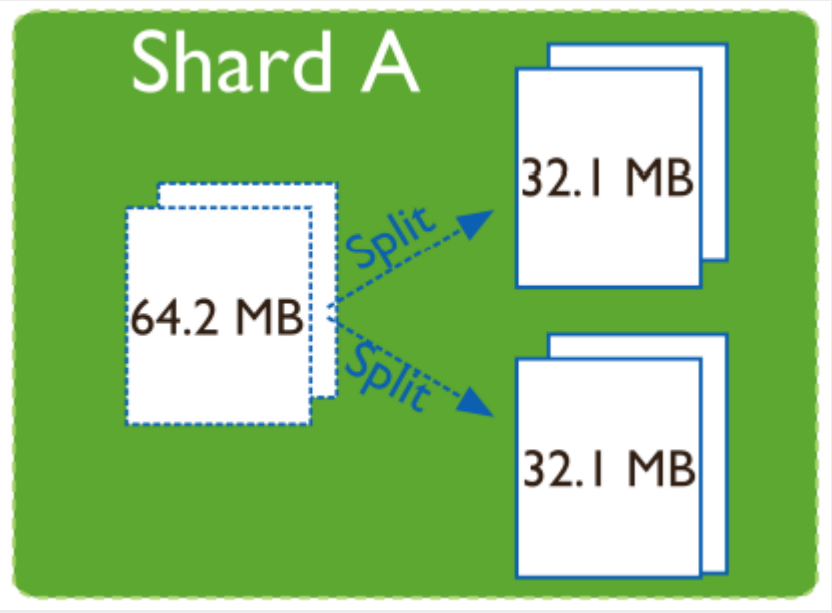
● Maintaining a Balanced Data Distribution

- ✓ mongoDB 는 두가지 background process 를 사용하여 균등 cluster 를 보장
  - ✓ Splitting
  - ✓ Balancer

1. mongoDB sharding 개요 – Balanced Data Distribution

● Splitting

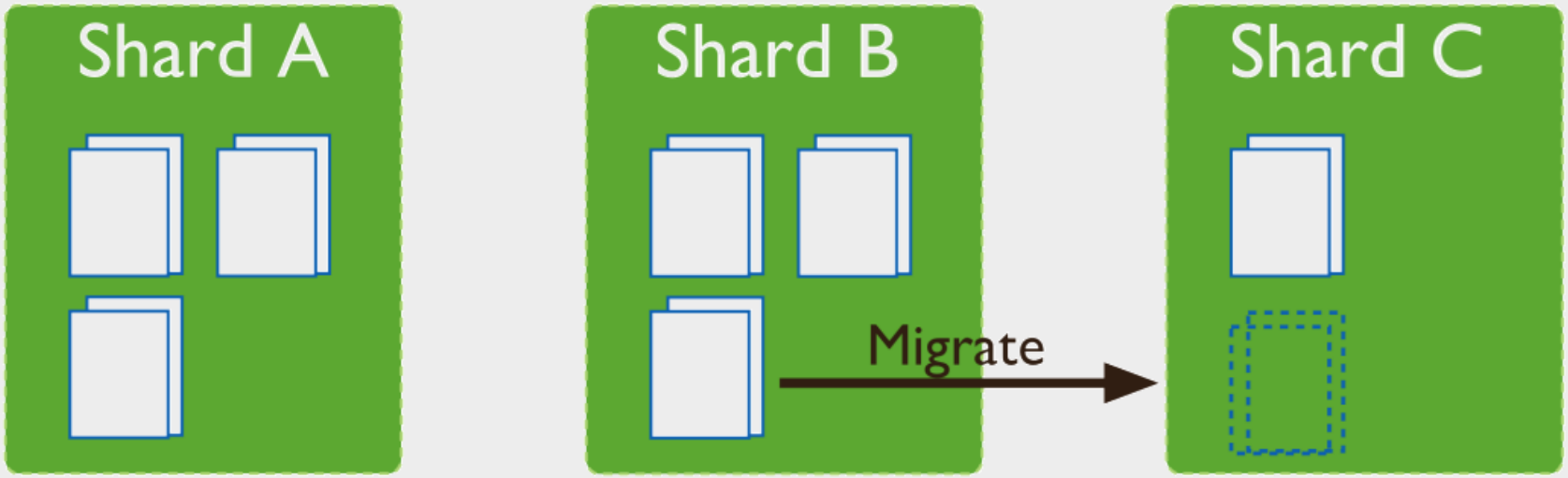
- ✓ Chunk 가 매우 커지는 것을 처리하기 위한 백그라운드 프로세스
- ✓ 특정 chunk size (64MB) 를 넘어서 커지기 시작하면 mongoDB 는 chunk 를 반으로 나눔
- ✓ Split 를 하기 위해 mongoDB 는 어떤 data 를 이동하거나 shard 에 영향을 주지 않음



1. mongoDB sharding 개요 – Balanced Data Distribution

● Balancing

- ✓ Chunk 이동을 관리하는 백그라운드 프로세스.
- ✓ Balancer 는 cluster 의 모든 query router 에서 실행된다.
- ✓ Cluster 의 sharded collection 의 분배가 불균등할때 balancer 가 많은 수의 chunk를 가진 shard 에서 적은 수의 chunk 를 가진 shard 로 collection 이 균형을 이룰 때까지 이동.



1. mongoDB sharding 개요 – How to route Reads and Writes

● Config servers

- ✓ config database 안에 각 데이터가 어떤 shard 에 위치해 있는지에 대한 metadata 를 저장한다.

● mongos instances (query router)

- ✓ config database 에 저장된 metadata 를 cache 하고 있다가 요청이 오면 Reads 와 Writes 를 shard 로 보내기 위해 사용한다.

2. mongoDB auto sharding 의 문제점 – background process

● Splitting

- ✓ Chunk 가 매우 커지는 것을 처리하기 위한 백그라운드 프로세스
- ✓ 특정 chunk size (64MB) 를 넘어서 커지기 시작하면 mongoDB 는 chunk 를 반으로 나눔
- ✓ Split 를 하기 위해 mongoDB 는 어떤 data 를 이동하거나 shard 에 영향을 주지 않음

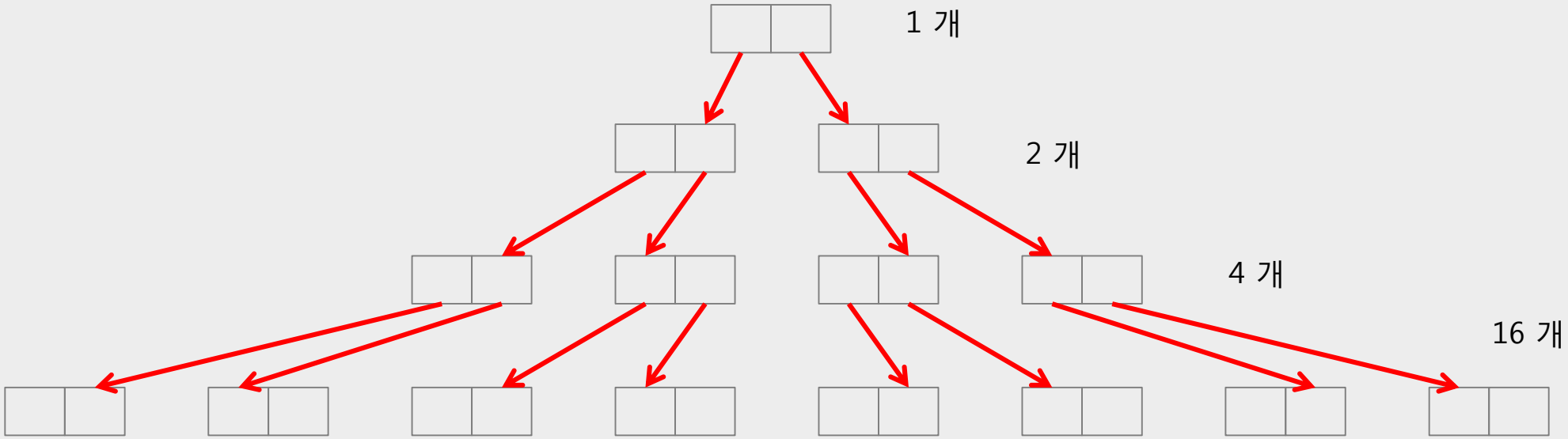
● Balancer

- ✓ Chunk 이동을 관리하는 백그라운드 프로세스.
- ✓ Balancer 는 cluster 의 모든 query router 에서 실행된다.
- ✓ Cluster 의 sharded collection 의 분배가 불균등할때 balancer 가 많은 수의 chunk를 가진 shard 에서 적은 수의 chunk 를 가진 shard 로 collection 이 균형을 이룰 때까지 이동.

2. mongoDB auto sharding 의 문제점 – background process

●Splitting 의 부하

✓ Data 량이 늘어 남에 따라 chunk 의 수가 기하 급수적으로 늘어날 수 있음



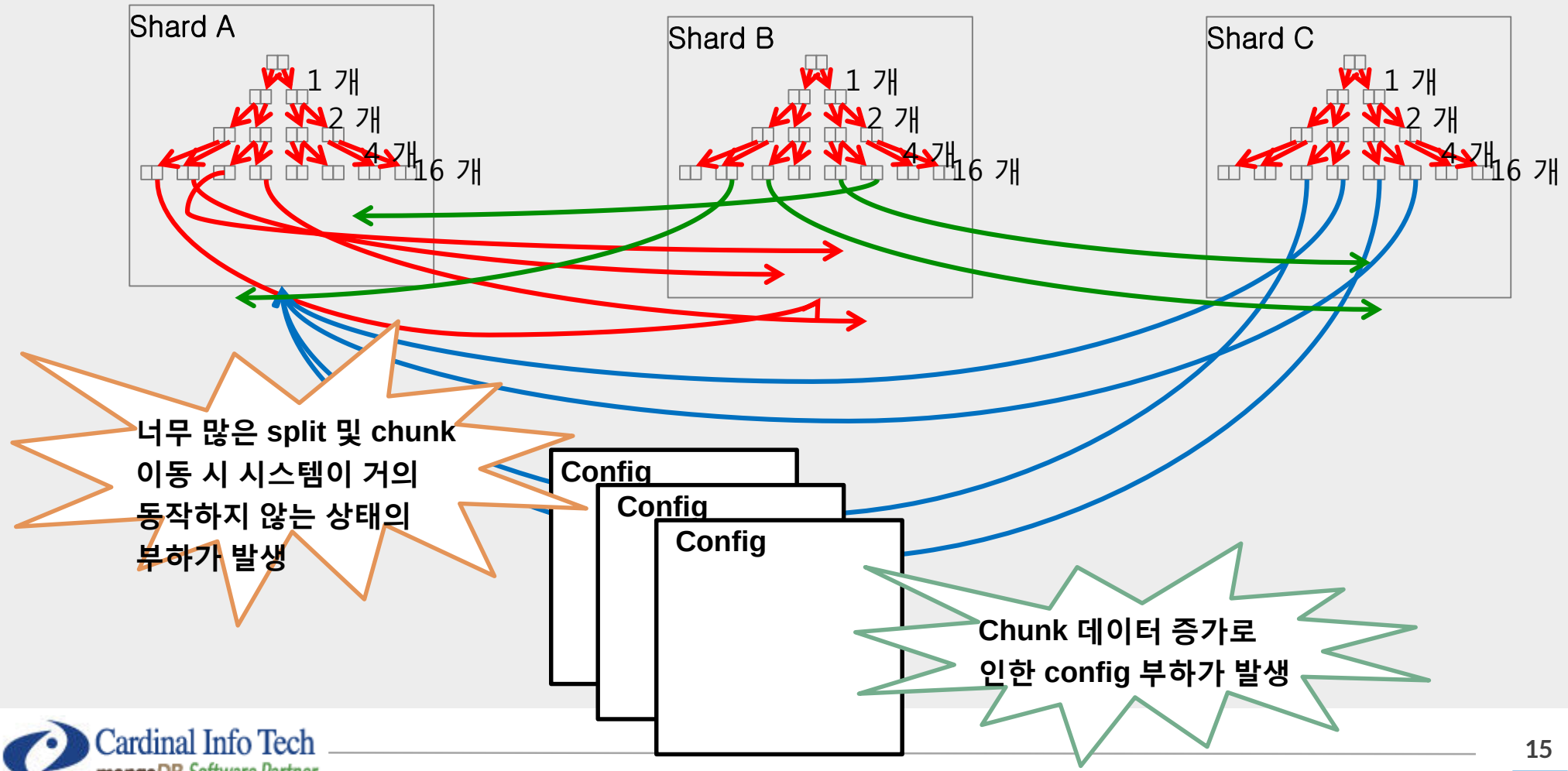
Splitting…Splitting…

Split 계산 시
30 초 소요

2. mongoDB auto sharding 의 문제점 – background process

● Balancing (moveChunk) 의 부하

✓ Chunk 량이 늘어 남에 따라 shard 의 chunk 개수의 balancing 을 맞추기 위해 move chunk 수행



2. mongoDB auto sharding 의 문제점 – background process

●Config server 의 부하

- ✓ Writes Data
  - ✓ 존재하는 chunks 를 쪼갤 때
  - ✓ Shard 간 chunk 를 옮길 때
- ✓ Reads Data
  - ✓ 최초 새로운 mongos 가 시작되거나 존재하는 mongos 가 재 기동 되었을 때
  - ✓ Chunk 를 옮긴 후, 모든 mongos 가 새로운 cluster 메타데이터로 갱신 될 때
- ✓ Distribution Lock 관리

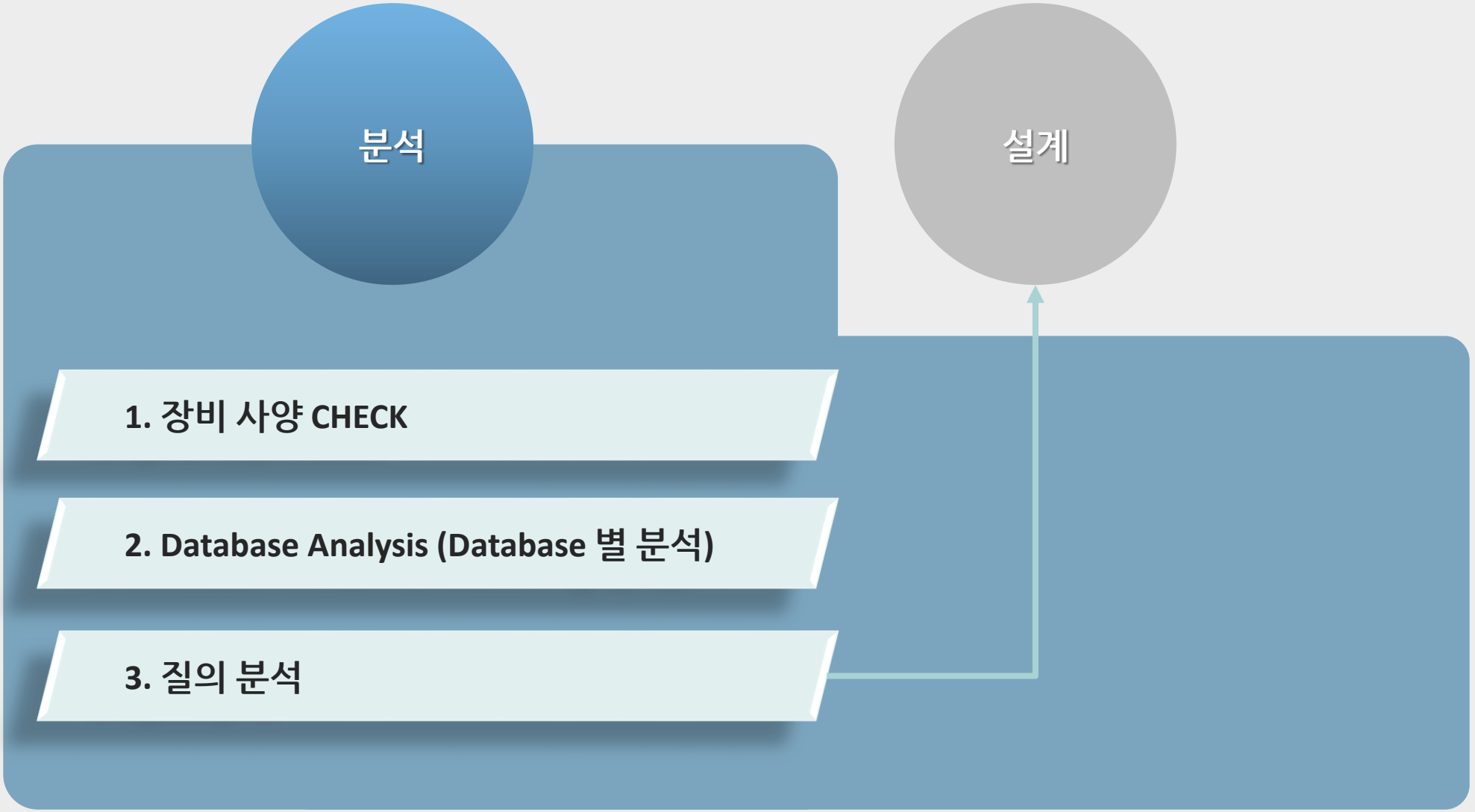
2. mongoDB auto sharding 의 문제점

- auto sharding...?

대규모 분산 시스템에서 auto sharding 만이 답인가?

NO

3. 전략 방안 – 분석



3. 전략 방안 – 설계

분석

설계

1. Insert DB 분리

2. Update DB 분리

Storage 중심 설계

- ✓ 대용량 스토리지 장비로 Shard 구성
- ✓ Out of memory 상태의 최대 성능이 결정 됨.
  ex) Out of memory 상태의 최대 성능이 3,000 OPS 이고 요구 사항이 6,000 OPS 라면 Insert DB를 2개의 Shard로 구성 – 장비 성능 측정 선행 필요
- ✓ 향후 증설도 동일한 방법으로 max 치 대비 증설

3. 전략 방안 – 설계

분석

설계

1. Insert DB 분리

2. Update DB 분리

In-Memory 중심 설계

- ✓ Update DB 가 In-Memory 가능하도록 Shard 구성
- ✓ 데이터베이스 크기를 가용 메모리 크기로 결정.
- ✓ Update DB size 에 따라 Shard 개수를 결정
- ✓ 향후 증설도 동일한 방법으로 DB size 에 따라 Shard 증설

3. 전략 방안 – 설계

분석

설계

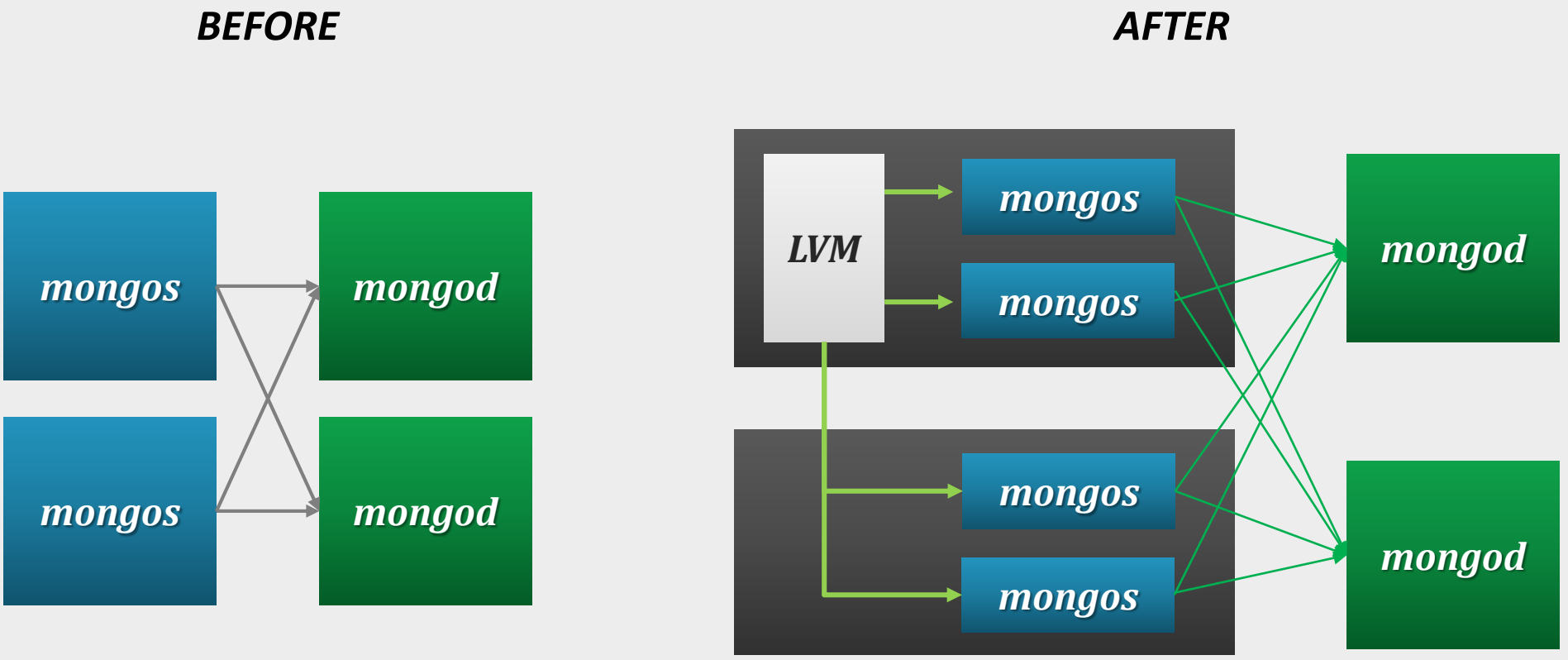
1. Insert DB 분리

2. Update DB 분리

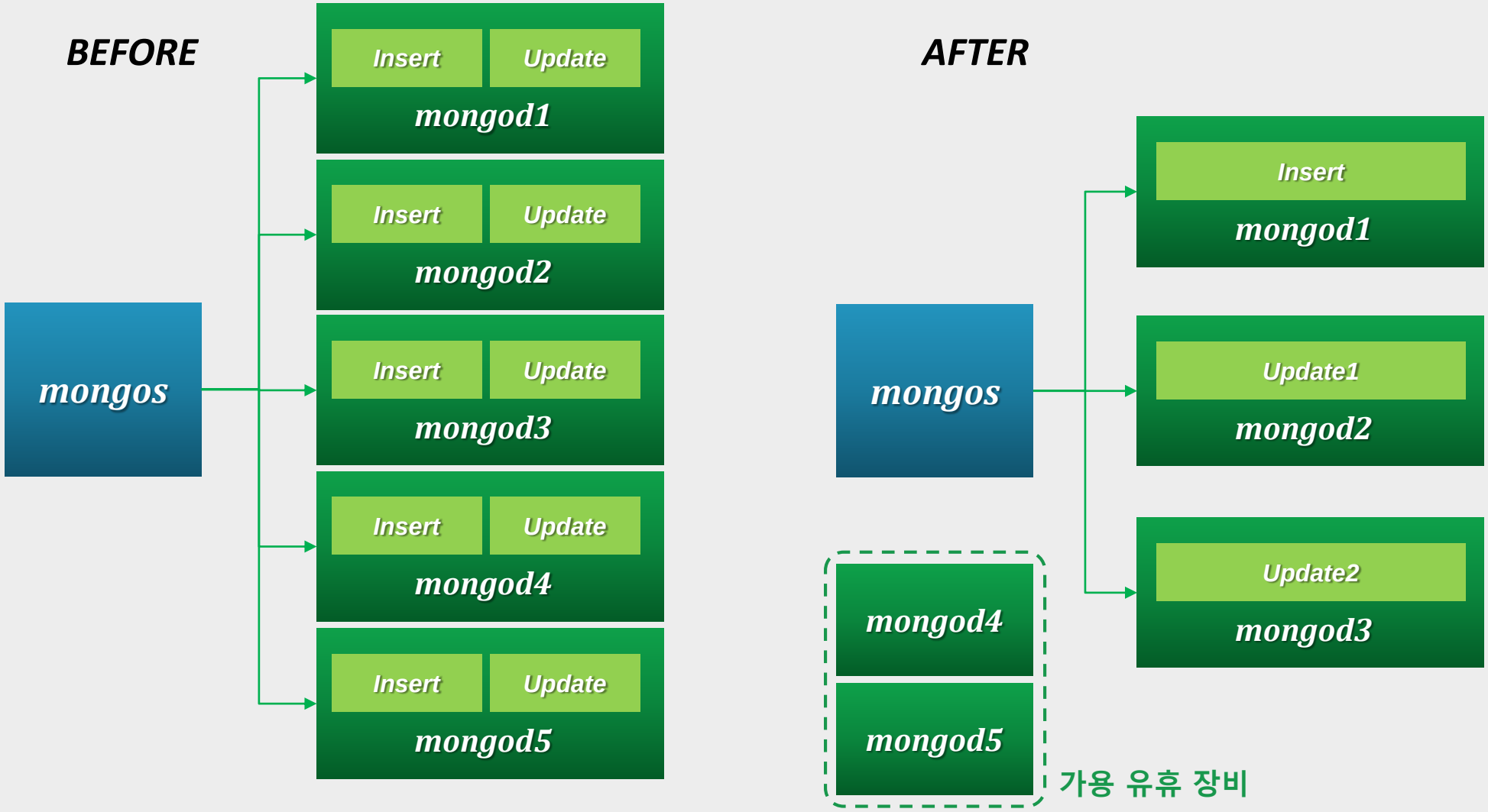
In-Memory 를 위한 날짜 별 DB 전략

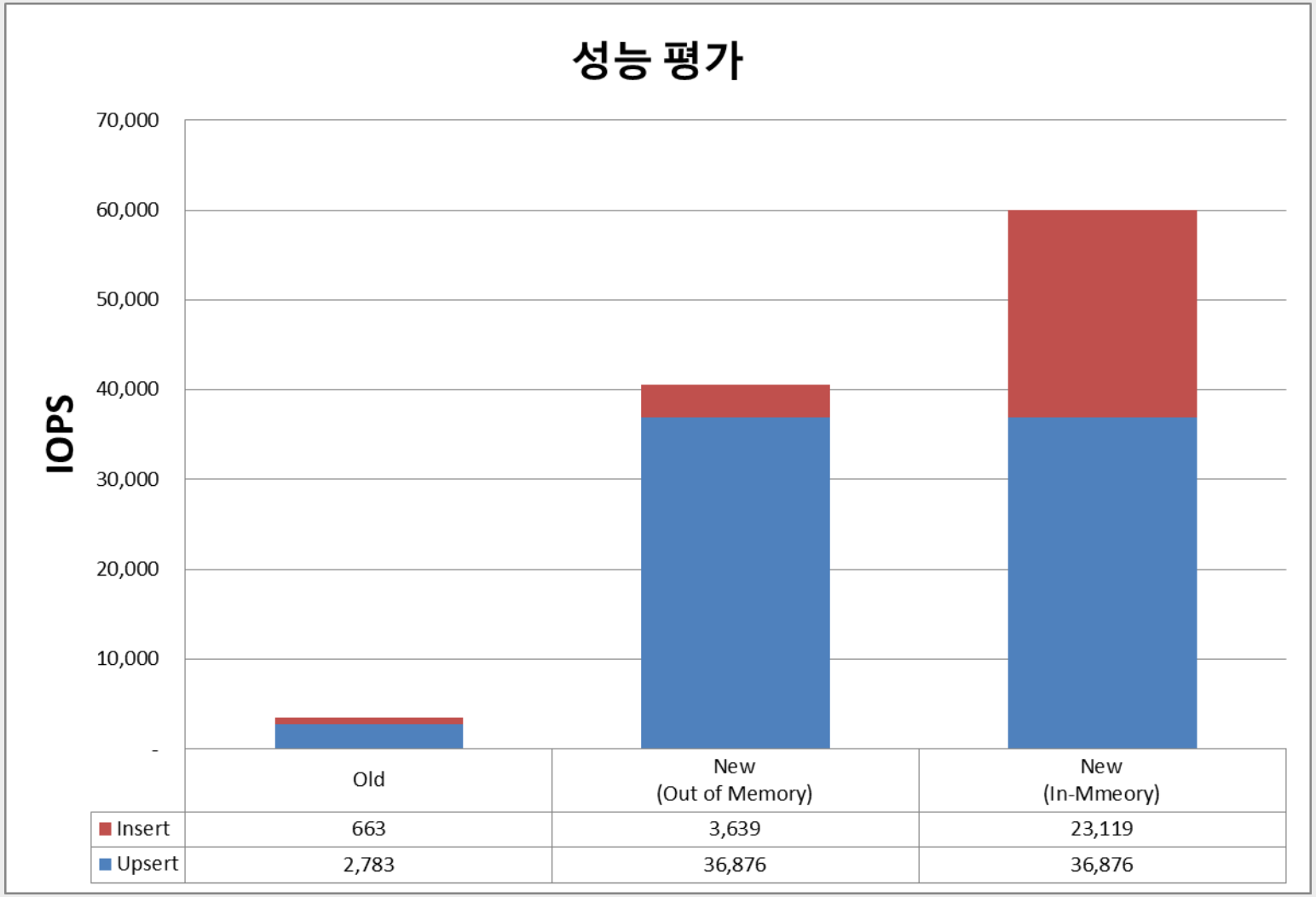
- ✓ 최근에 업데이트 되는 데이터는 LRU 방식으로 관리
- ✓ In-memory 유지를 위한 최소 데이터 양을 결정
  ex) 1일 단위 데이터 양
- ✓ In-Memory를 유지할 수 있는 work-set 설정
- ✓ Aggregation 기반의 배치 처리를 secondary 에서 처리하여 부하를 줄임

4. 사례 연구 – Case 1


BEFORE
mongos
mongos
mongod
mongod
AFTER
LVM
mongos
mongos
mongos
mongos
mongod
mongod

4. 사례 연구 – Case 2


BEFORE
mongos
Insert
Update
mongod1
Insert
Update
mongod2
Insert
Update
mongod3
Insert
Update
mongod4
Insert
Update
mongod5
AFTER
mongos
Insert
mongod1
Update1
mongod2
Update2
mongod3
mongod4
mongod5
가용 유휴 장비



	Old	New (Out of Memory)	New (In-Mmeory)
Insert	663	3,639	23,119
Upsert	2,783	36,876	36,876