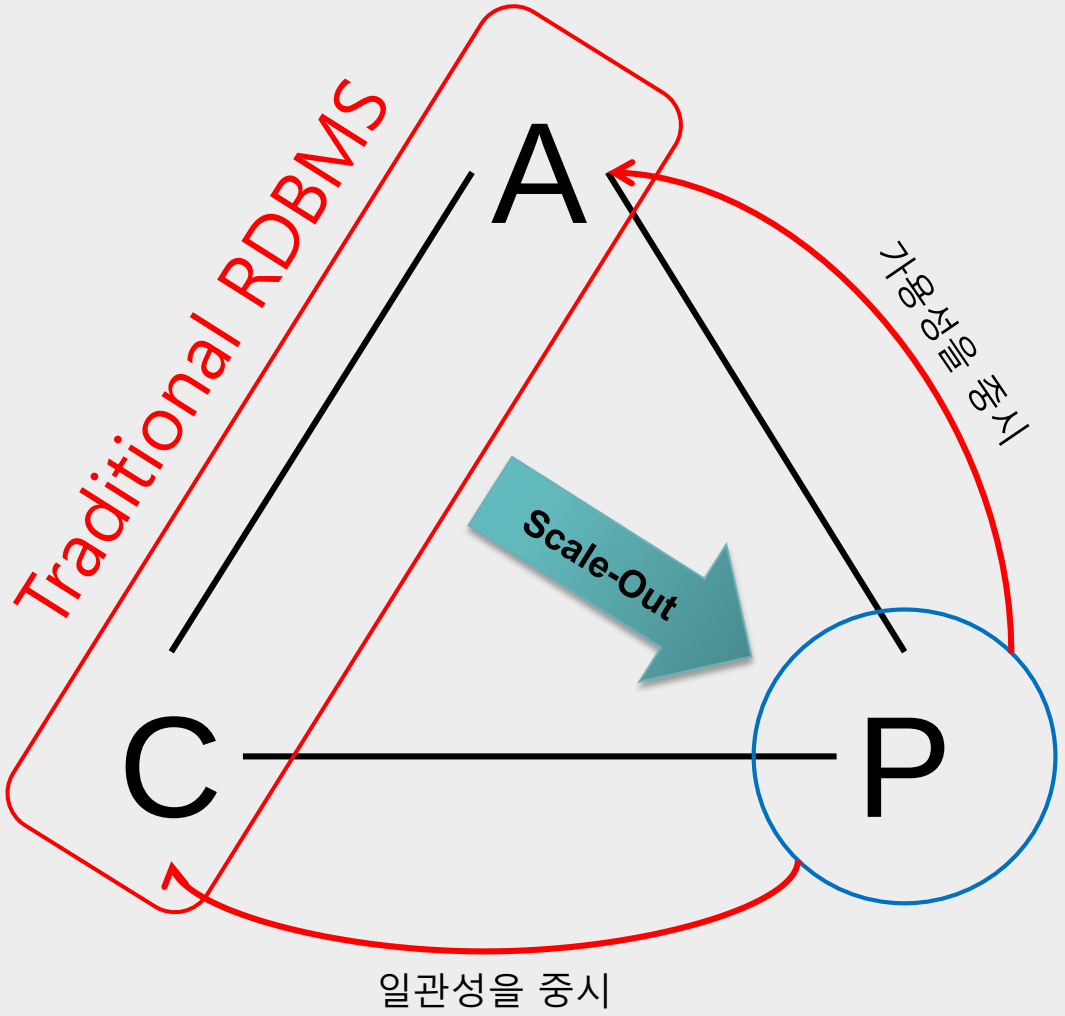
결과적 트랜잭션 및 트랜잭션 로그를 이용한 MongoDB 트랜잭션 처리방법

AGENDA

1. 분산시스템 개요
2. MongoDB의 트랜잭션
3. MongoDB를 위한 트랜잭션 처리 기술
4. 마치며.....

1. 분산시스템 개요 - CAP 이론



**트랜잭션을 포기하고,
대용량 데이터를 처리하기
위해 확장성을 보장한
데이터베이스가
NoSQL이다.**

1. 분산시스템 개요 – ACID vs. BASE

RDB의 ACID

Atomic
Consistent
Isolation
Duration

NoSQL의 BASE

Basically
Available
Soft state
Eventual
Consistency

1. 분산시스템 개요 – BASE의 개념

● Basically Available

- ✓ 분산 시스템을 기반으로 하기 때문에, 가용성은 보장되어야 한다.
- ✓ 장애가 발생하더라도 사용자는 서비스를 통한 데이터 액세스가 보장되어야 한다.

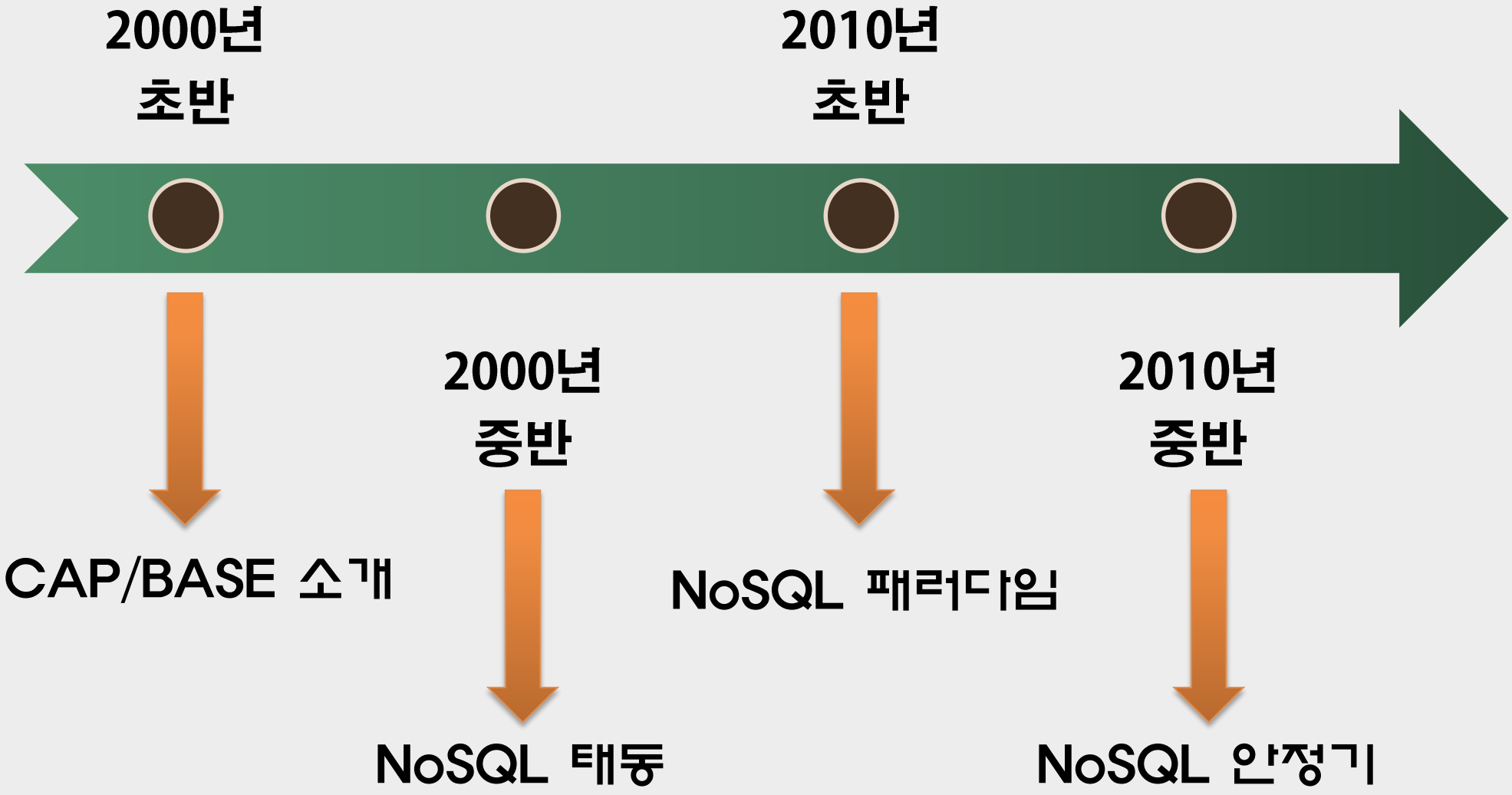
● Soft State

- ✓ 상태는 확률적 또는 명시적으로 유지된다.
- ✓ 상태는 일관성이 필수적인 것은 아니다. -> 유연한 또는 약한 일관성
- ✓ 사용자와 네트워크는 상태를 유지하기 위한 책임을 지지만, 궁극적으로 사용자가 모든 책임을 진다.

● Eventual Consistency

- ✓ 일시적이나마 무결성을 보장할 수 없지만, 일정 시간이 지나면 무결성을 보장할 수 있다.
- ✓ Soft State를 이용하여 네트워크가 살아 있다면, 일관성 정책은 사용자가 결정한다.

1. 분산시스템 개요 – NoSQL의 발자취



2. MongoDB의 트랜잭션

● Atomic

- ✓ 한 개의 document에 대한 원자성을 보장
- ✓ 장애 발생시 복수개의 row를 처리하는 한 개의 연산에 대한 원자성은 보장하지 않음.

● Consistent

- ✓ Master에서만 읽기/쓰기를 수행할 경우 일관성 유지
- ✓ Master에서 쓰기, Slave에서 읽기를 수행하는 복제 구조에서는 write concern을 이용한 일관성 정책

● Isolation

- ✓ 지원 Isolation Level - Read Uncommitted 레벨 적용

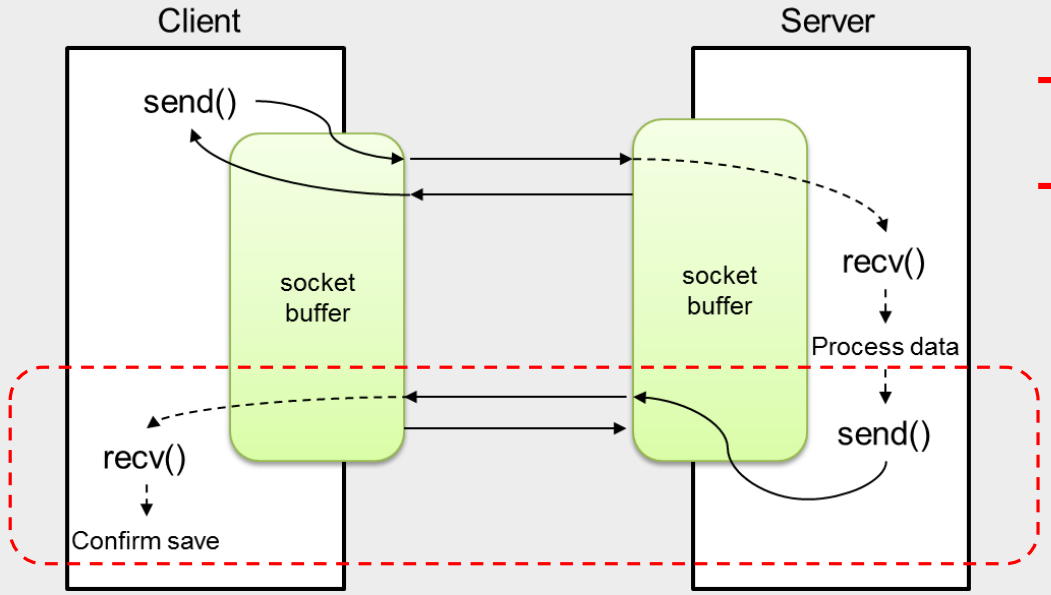
● Duration

- ✓ 단일 노드 : journaling write concern, flush write concern
- ✓ 복제 노드 : 복제 개수 write concern, 과반수를 위한 majority

2. MongoDB의 트랜잭션 – RDBMS vs. MongoDB

	RDBMS	MongoDB
● Atomic	✓ 복수개의 Row 또는 연산 보장	✓ 단일 document에 대한 보장
● Consistent	✓ 복제 시스템에서는 복제 완료까지 대기	✓ Write concern을 통한 정책 설정
● Isolation	✓ Read Uncommitted/Committed ✓ Repeatable Read ✓ Serializable	✓ Read Uncommitted
● Duration	✓ Commit	✓ Write concern을 통한 정책 설정

2. MongoDB의 트랜잭션 – MongoDB의 write concern



None

Normal

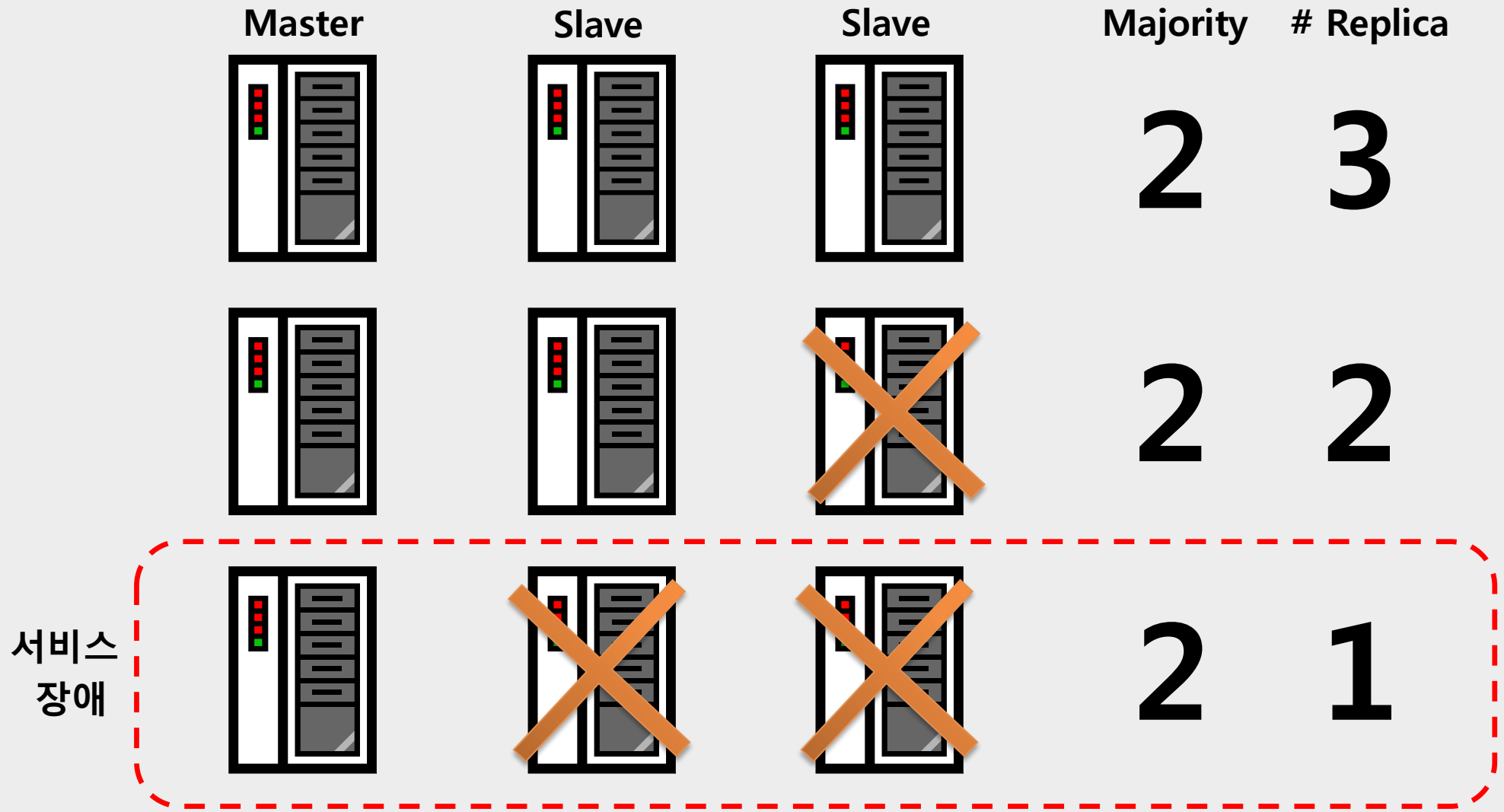
getLastError() → Safe

2. MongoDB의 트랜잭션 – MongoDB의 write concern

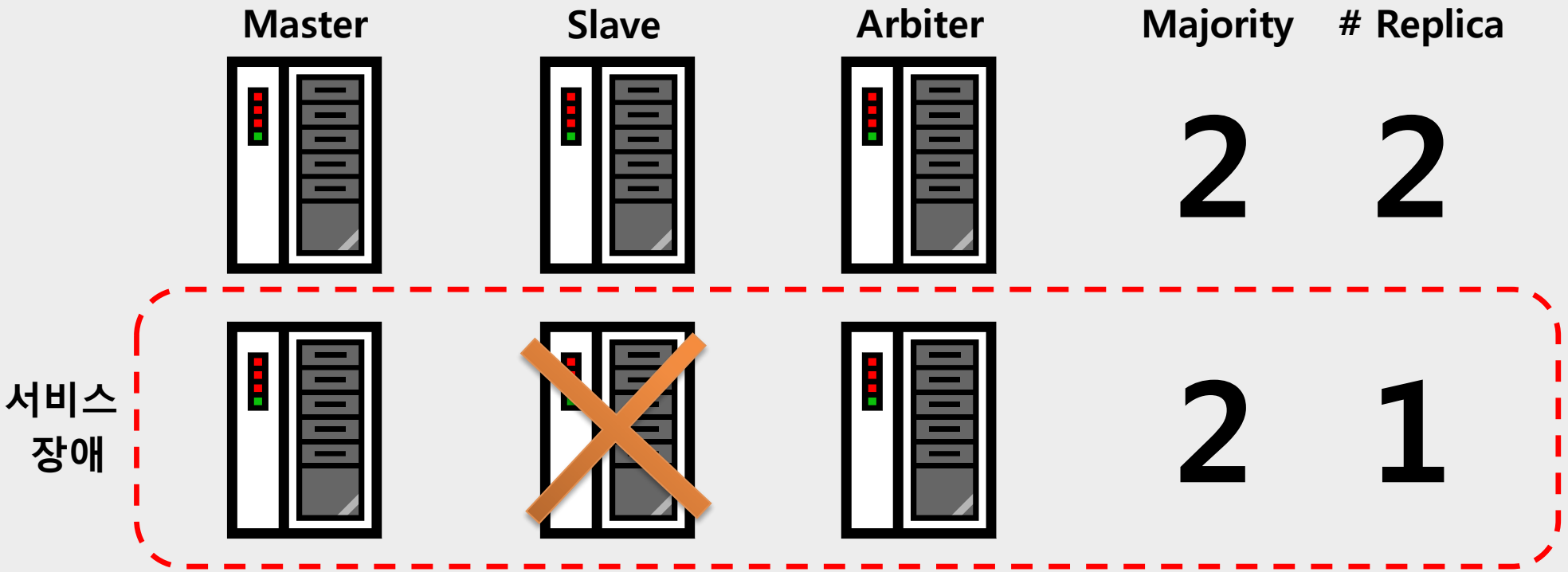
Write Concern	None	Normal(w=0)	Safe(w=1)	Replica(w=2)
성능	100%	100%	62%	46%

● 데이터 안정성을 위해 Consistent와 Duration을 보장받기 위해서는 그만한 성능 저하가 뒤따라 온다.

2. MongoDB의 트랜잭션 – MongoDB의 Majority



2. MongoDB의 트랜잭션 – MongoDB의 Majority



2. MongoDB의 트랜잭션 – MongoDB 트랜잭션 해결 과제

● Atomic

- ✓ 복수개의 document에 대한 원자성을 보장할 수 있어야 한다.
- ✓ 트랜잭션으로 구성된 복수개의 document는 HDD에 동시에 쓰여지거나 또는 쓰여져서는 안 된다.

● Consistent

- ✓ 복제 시스템의 일관성은 복제 구성원 장애에 의해 동적으로 변동될 수 있어야 한다.

● Isolation

- ✓ 트랜잭션이 제공하는 Isolation Level을 제공할 수 있어야 한다.

● Duration

- ✓ 사용자 선택에 따라 write concern을 사용하는 방법과 복제 구성원 수를 증가하는 방법으로 해결 가능

3. MongoDB 트랜잭션 처리 기술

● Atomic

- ✓ 트랜잭션을 구성하는 복수개의 document가 commit될 때, flush가 발생하지 않도록 Lock을 제공
- ✓ 트랜잭션 복구를 위한 2-Phase Commit 형태의 **결과론적 트랜잭션(Eventual Transaction)**

● Consistent

- ✓ Majority의 값을 복제 구성원 장애가 발생할 때 마다 변동할 수 있는 **Dynamic Majority**를 제공

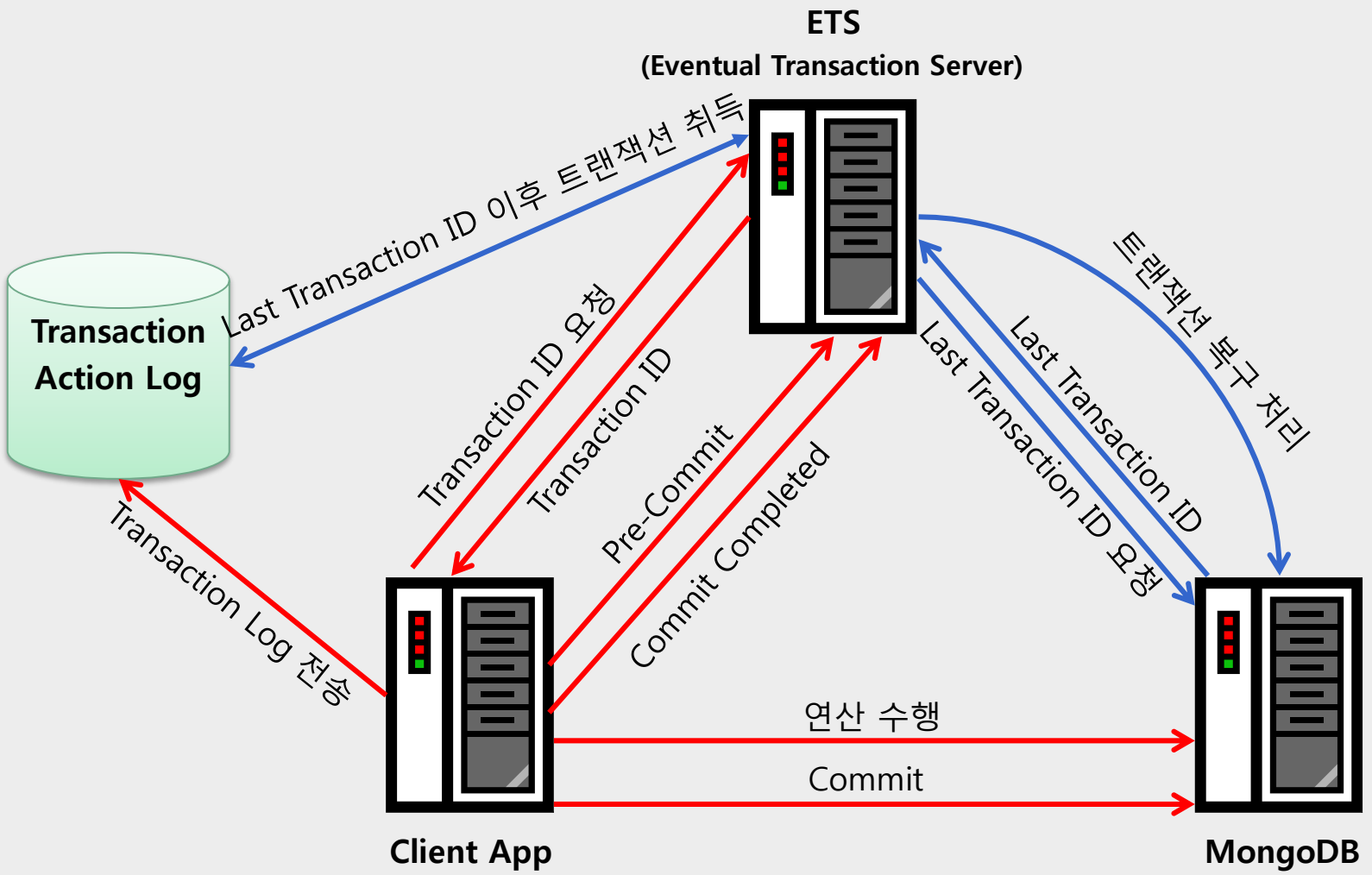
● Isolation

- ✓ **트랜잭션 로그와 분산 트랜잭션 서버**를 이용한 Transaction Isolation Level의 다양화
  - ➢ Read Uncommitted
  - ➢ Read Committed
  - ➢ Repeatable Read
  - ➢ **Preemptive Lock**
  - ➢ **Channel Serializable**
  - ➢ **Named Lock**

결과론적 트랜잭션(Eventual Transaction)

시스템 장애에 의해 트랜잭션이 결여될 수 있지만, 시스템 복원과 함께 트랜잭션 복구를 통해 트랜잭션 ACID를 보장하는 방법

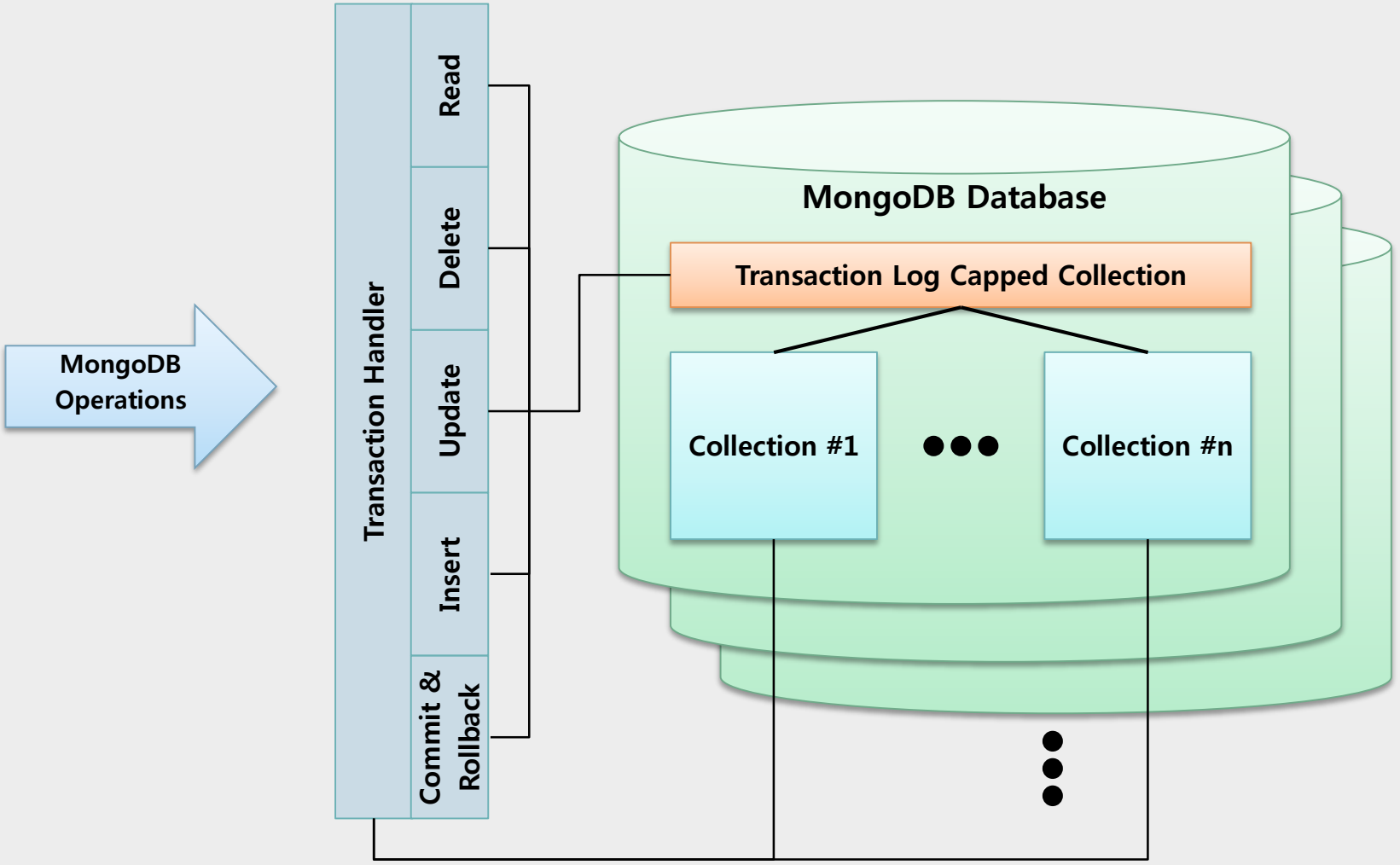
3. MongoDB 트랜잭션 처리 기술 – 결과론적 트랜잭션



트랜잭션 로그(Transaction Log)

MongoDB Database별로 존재하는 트랜잭션 수행결과를 저장하는 로그로 Capped Collection으로 구성되어져 있으며, 트랜잭션 commit 발생시 해당 트랜잭션 데이터를 하나의 연산으로 갱신한다.

3. MongoDB 트랜잭션 처리 기술 – 트랜잭션 로그



3. MongoDB 트랜잭션 처리 기술 – Isolation Level

● Level 0 : Read Uncommitted

- ✓ 다른 트랜잭션은 해당 트랜잭션이 변경한 document를 Commit 수행하기 전에 읽을 수 있다.
- ✓ 시스템 장애에 따른 원자성이 깨질 수 있다.

● Level 1 : Read Committed

- ✓ 다른 트랜잭션은 해당 트랜잭션이 변경한 document를 Commit 을 수행한 후에 읽을 수 있다.

● Level 2 : Repeatable Read

- ✓ 다른 트랜잭션은 해당 트랜잭션이 변경한 document를 Commit을 수행한 후에라도 Commit 이전의 데이터를 반복적으로 읽어 들인다. 만약, 변경된 document를 읽기 위해서는 새로운 트랜잭션을 수행하여야 한다.

● Level 3 : Preemptive Lock

- ✓ 해당 트랜잭션이 액세스(read or write)한 document는 다른 트랜잭션이 액세스할 수 없으며, 해당 트랜잭션이 commit을 수행한 후에 액세스가 가능하다.

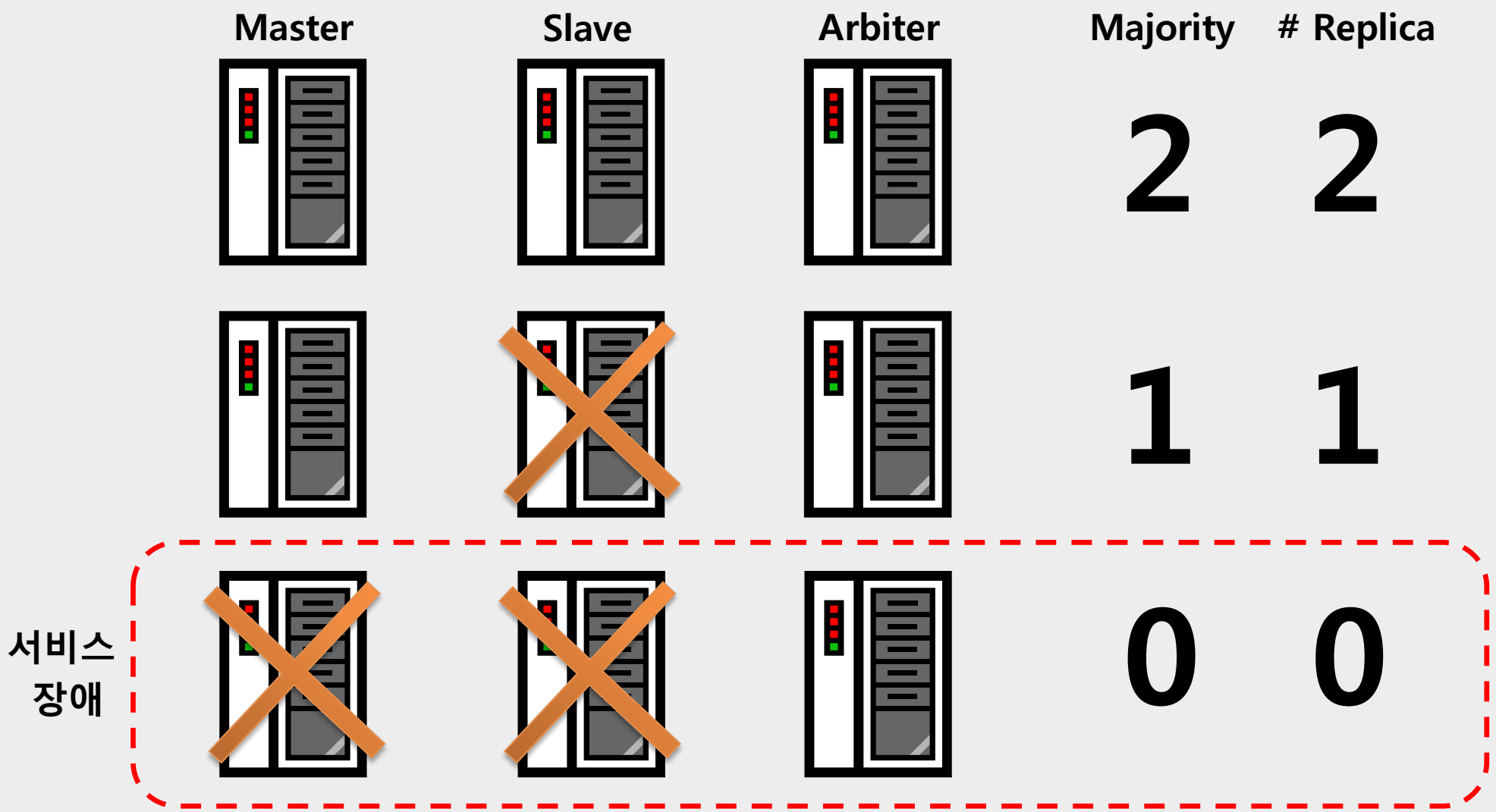
● Channel Serializable

- ✓ MongoDB의 Collection별로 채널을 구성하여 트랜잭션을 직렬화하여 처리한다.
- ✓ 채널간 트랜잭션의 동시성을 보장한다.

● Named Lock

- ✓ 분산 락(Distributed Lock)을 이용한 사용자 정의 트랜잭션

3. MongoDB 트랜잭션 처리 기술 – Dynamic Majority



3. MongoDB 트랜잭션 처리 기술 – Scalable Transaction

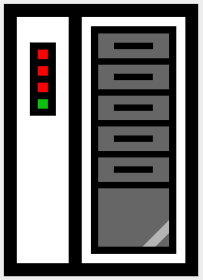
- 수평적 확대를 통한 트랜잭션 처리 능력이 증가하는가?

Concurrent Transaction을 보장하는가?

YES

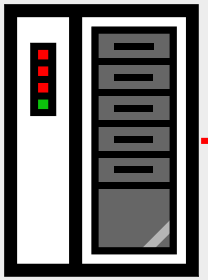
3. MongoDB 트랜잭션 처리 기술 – Scalable Transaction

ETS
(Eventual Transaction Server)

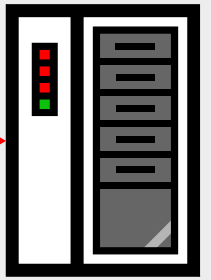


- Channel Serializable
- Name Lock

- Read Uncommitted
- Read Committed
- Repeatable Read
- Preemptive Lock



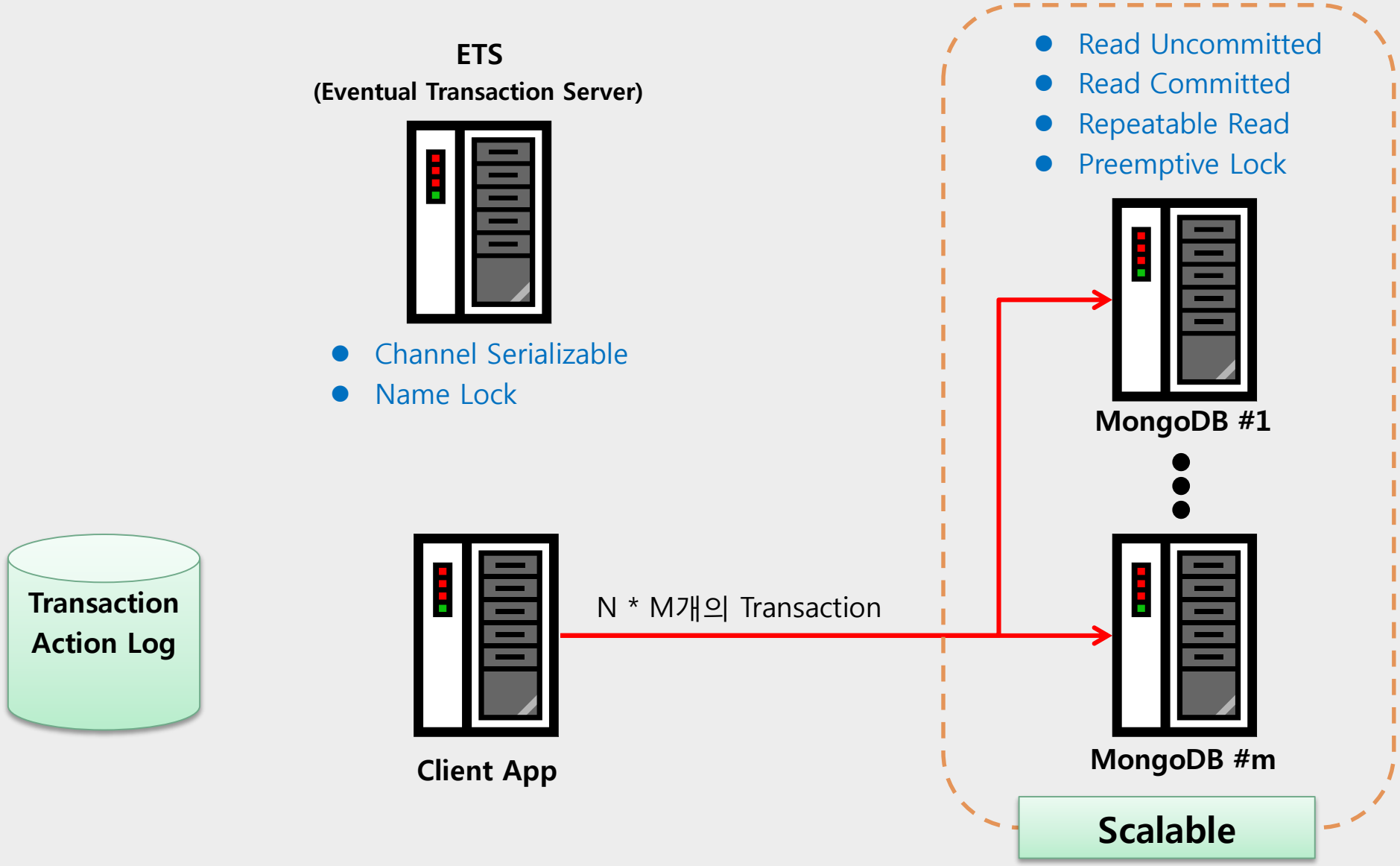
N개의 Transaction



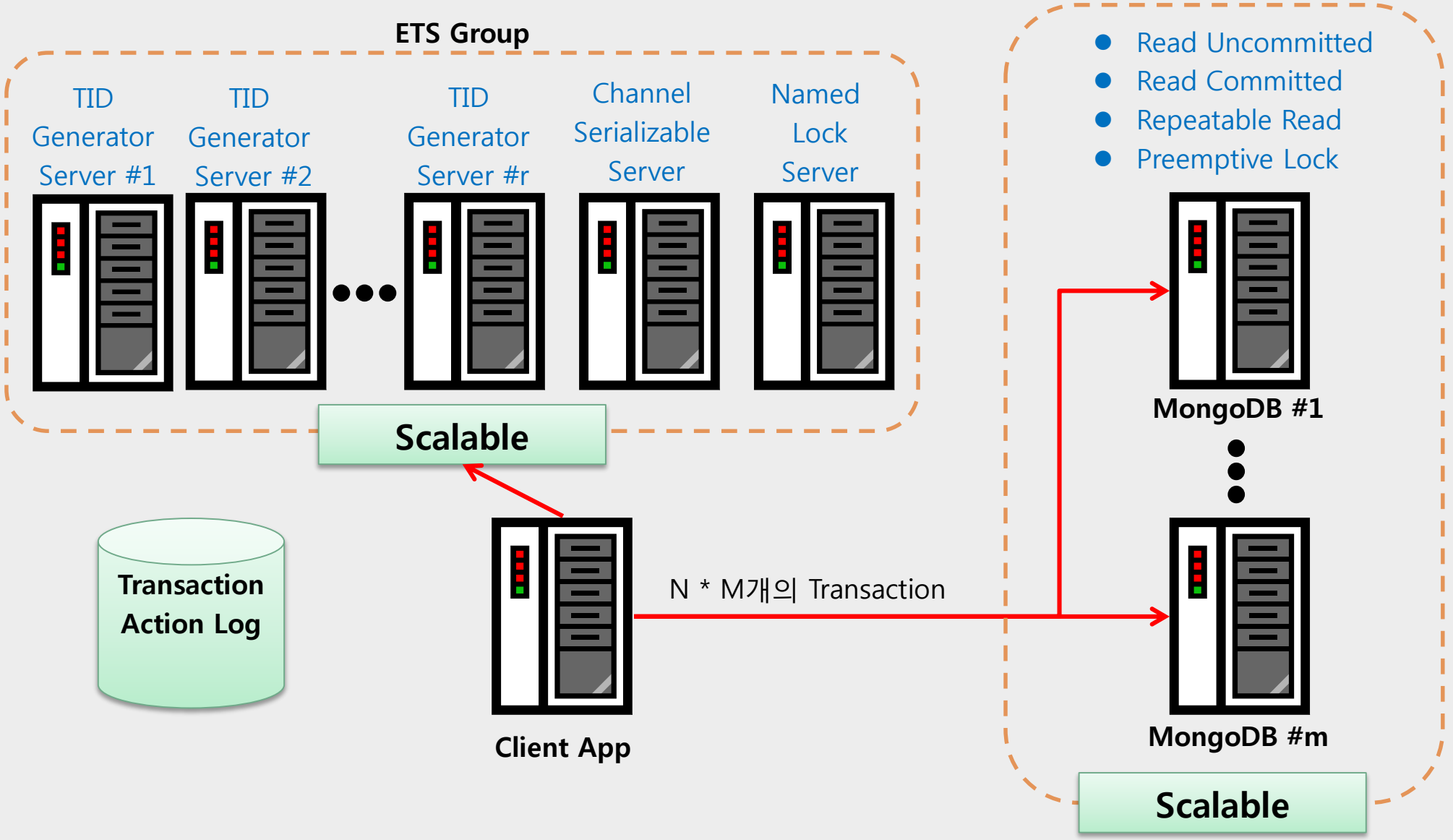
Client App

MongoDB

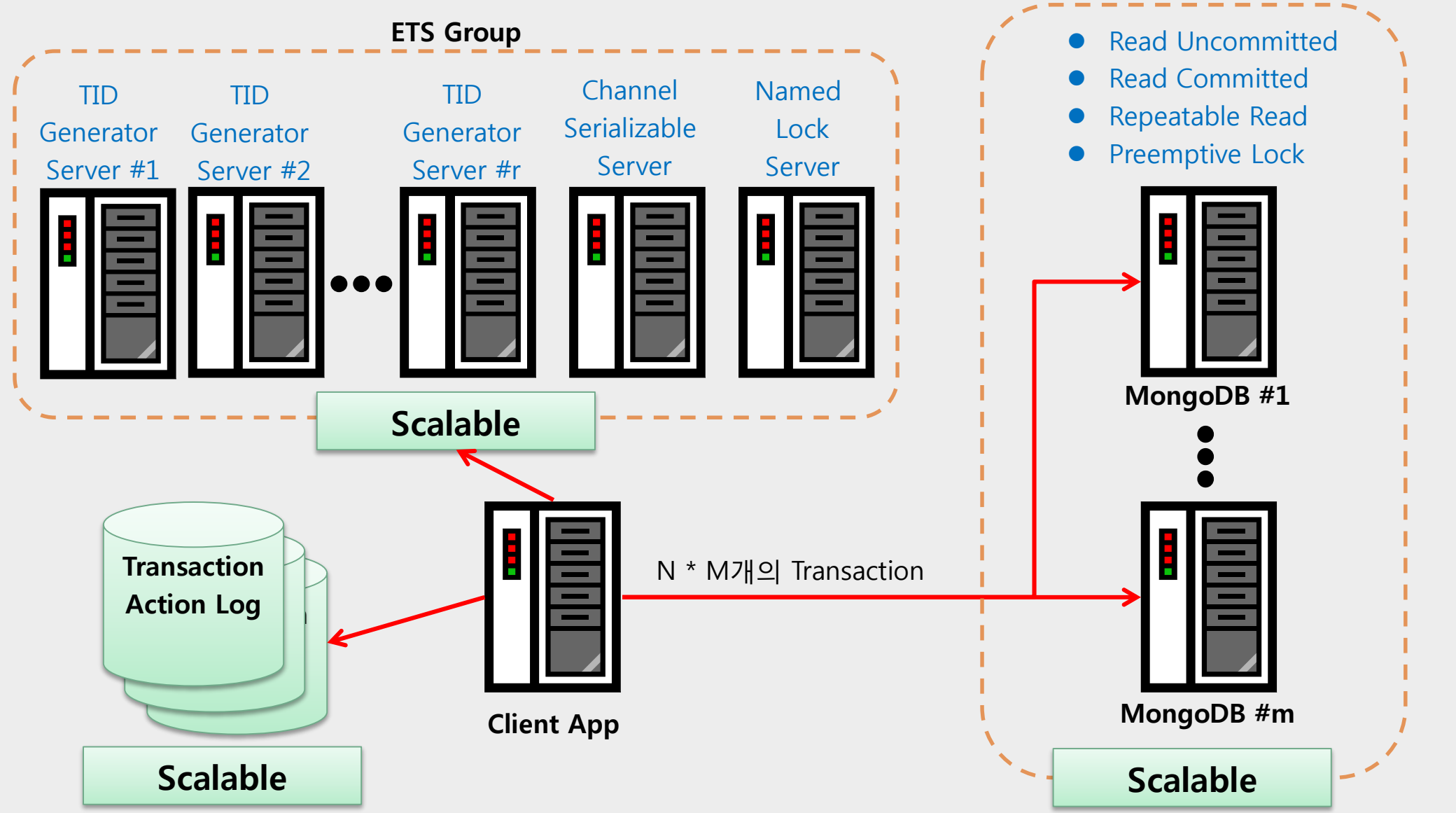
3. MongoDB 트랜잭션 처리 기술 – Scalable Transaction



3. MongoDB 트랜잭션 처리 기술 – Scalable Transaction



3. MongoDB 트랜잭션 처리 기술 – Scalable Transaction



4. 마치며 – MongoDB의 현주소

● 로그 데이터

- ✓ 데이터 분실이 발생하더라도 서비스에 큰 영향을 주지 않는 서비스
- ✓ 데이터를 적재하고 배치로 분석하는 서비스
- ✓ 형식이 일정하지 않는 트랜잭션 로그 데이터

● 약한 일관성

- ✓ 사용자 행위가 즉시 반영되기 보다는 일정시간 이후에 반영되어도 되는 서비스 – 사용자 댓글, 메시지

● 대용량 처리

- ✓ 데이터의 안정성을 위해 고가용성을 높이고, 대용량으로 발생하는 트래픽을 처리

● Profile Project

- ✓ 시범 서비스 또는 연구소 단위의 프로젝트에서 수행

4. 마치며 – 필요한 점

4. 마치며 – 필요한 점