RDB 확장성 (Scale Out) 어떻게 해결할 것인가 ?

조현기 (Jacob Jo)
jacob.jo@mariadb.com

MariaDB Korea Conference 2019.09.25

RDB 확장과 서비스 확장은 필수적 관계

> 서비스 관점

- 더 나은 사용자 경험
- 더 많은 사용자 처리
- 더 많은 거래의 처리

RDB 증가하는 서비스를 어떻게 수용해 왔는가 ?

> H/W Scale Up

- Memory (양적/질적)
- SSD
- NVMe
- More CPUs
- Better CPU Clock Speed
- Hyper-Threading
- Optimization and optimization ... w/ given & limited resources

RDB 잘 쓰기 – 가장 기본적이고 중요한 것 부터

> 기본에 충실하게
> 아래의 것만 충족되어도 되어도 어지간한 서비스는 날아다닙니다

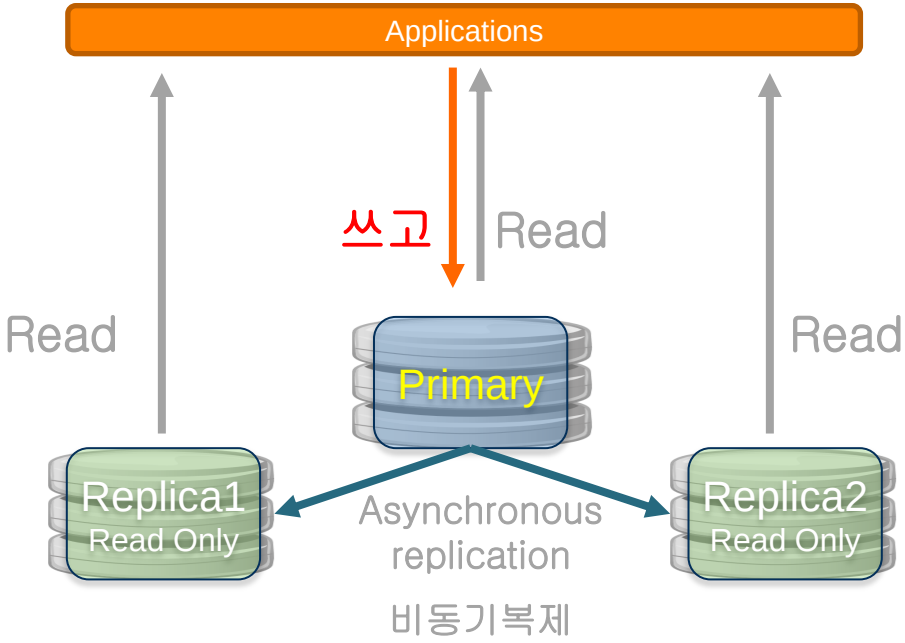
- 서비스 이해
- 데이터베이스에 적합한 인프라 선택 구성
- 데이터 모델 설계 (스키마/테이블/PK 설계)
- 최적화된 인덱스 설계
- 성능 효율적인 SQL 사용
- 어플리케이션 개발 초기 단계 부터 협업/커뮤니케이션
- 기타 등등

A. 가장 많이 사용되는 확장의 방법

Primary and Read Only Replicas

MariaDB

Primary and Read Only Replicas



Primary and Read Only Replicas

Binary Log Position
GTID Position

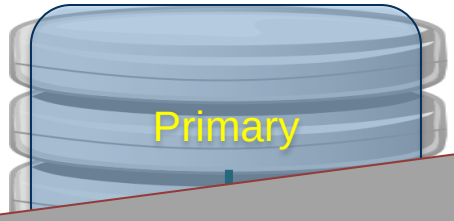
Transaction Log (Redo)

how often the transactions are flushed to the redo log
- Innodb Redo Log innodb_flush_log_at_trx_commit
  1 each trasaction performed
  0 once a second
- innodb_flush_method

*Semi-synchronousreplication
Acknowledge back to Primary (ACK Receiver Thread)
right after writing Relay Log is done
2

The binary log contains a record of all changes to the databases, both data and structure.
It consists of a set of binary log files and an index.

Primary and Read Only Replicas



*Asynchronous
replication

순서를 보장하기 위한
Threaded
.0.5 이후
a pool of
Threads
for
lication

xecute

Bina
G

how ofte
- Innodb
1 each
0 once a second
- innodb_flush_method

Primary (ACK Receiver Thread)

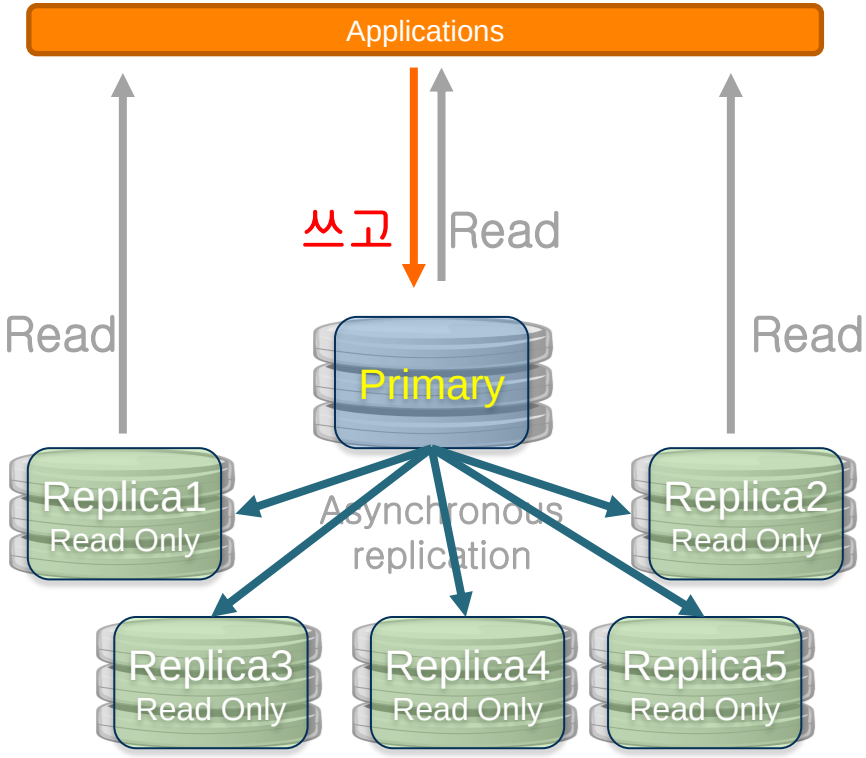
ght after writing Relay Log is done

2

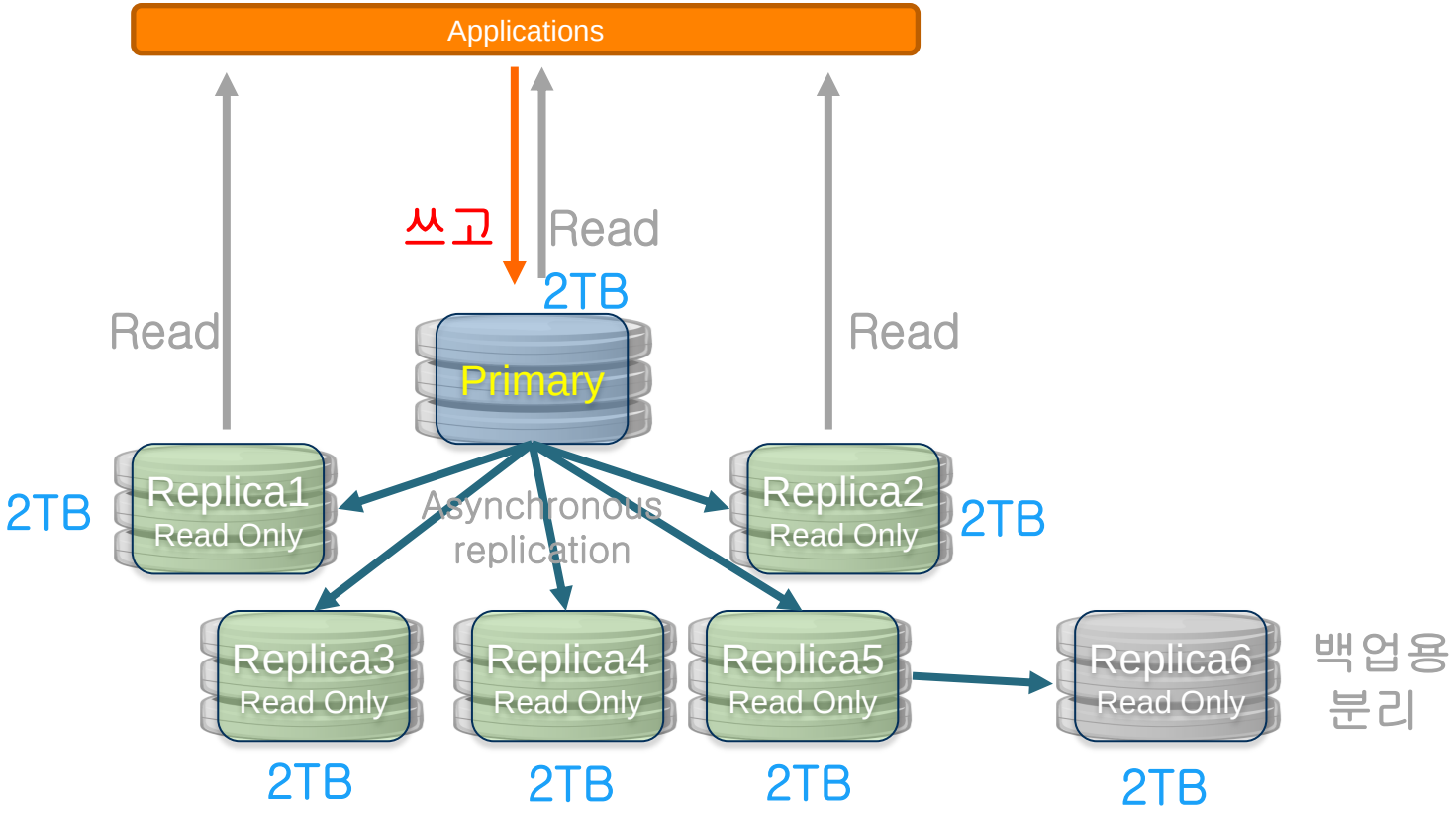
Primary-Replica 데이터반영
시간갭 존재

The binary log contains a record of all changes to the databases, both data and structure.
It consists of a set of binary log files and an index.

Primary and Read Only Replicas



Primary and Read Only Replicas

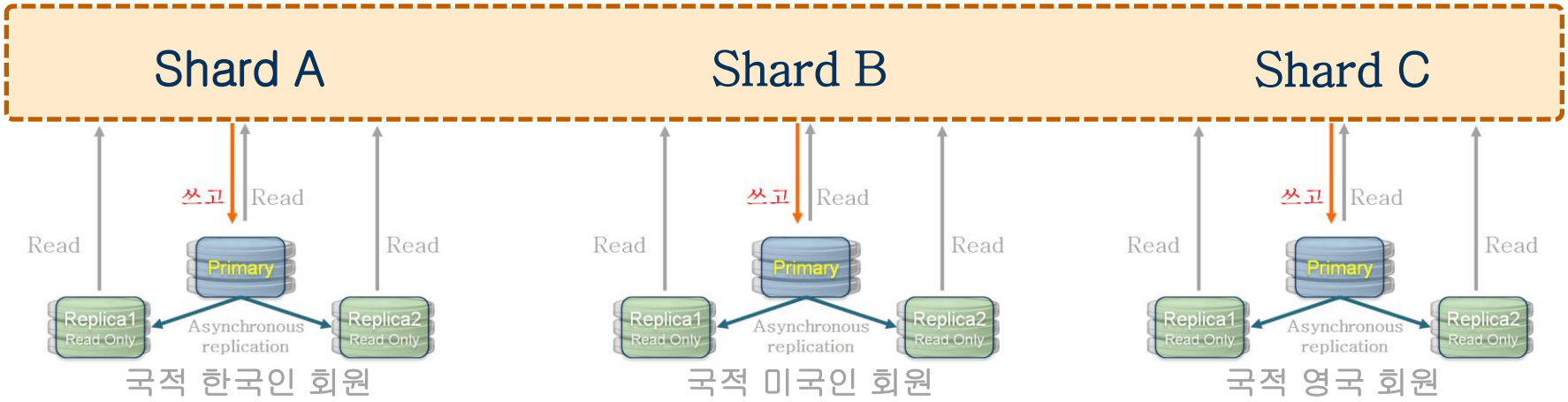


B. 또 다른 확장의 방법

샤딩과 업무영역별 분리

MariaDB

샤딩

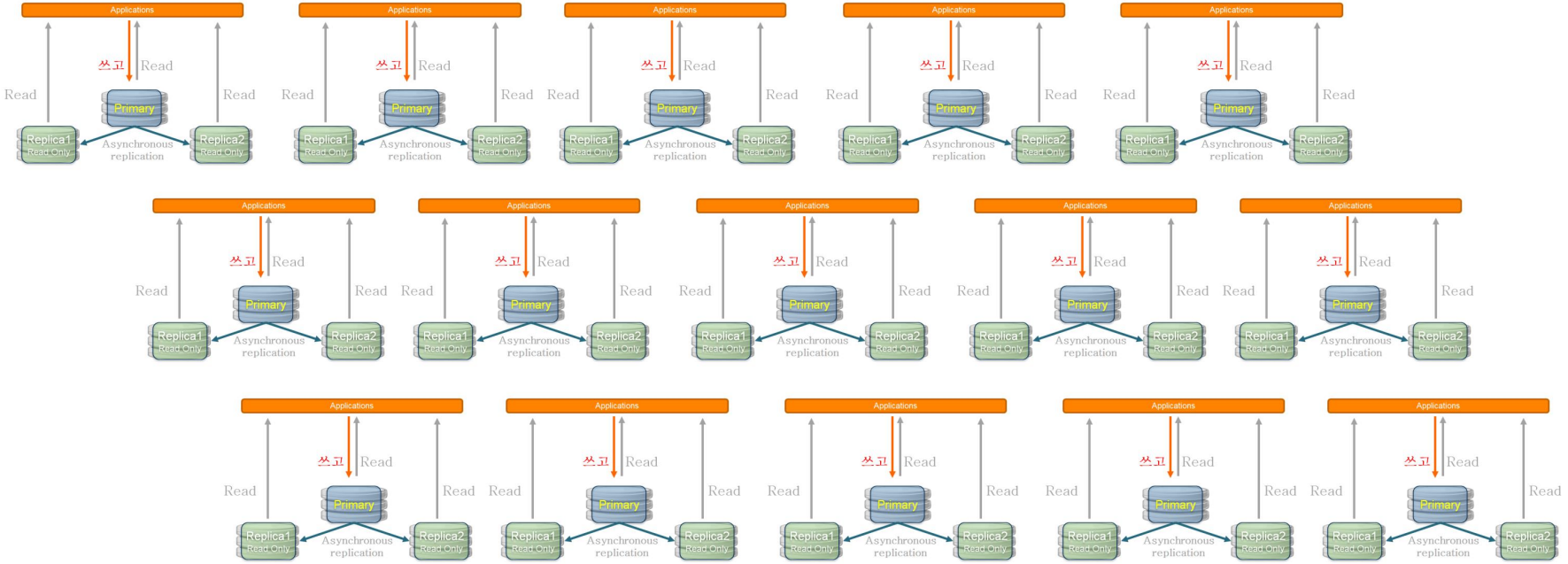


- Shard Key 설계 및 관리 복잡 부담 증가
  예) 한 테이블의 Row 를 회원의 국적코드별로 나누어(Shard) 저장
- 특정 샤드로데이터 쏠림(Skewnewss) 현상 발생 및 후 조치 어려움
- 변경이 없고 데이터와 부하를 골고루 분산시킬 수 있는 Shard Key 선정

업무영역별 분리

여러개의 Primary 를 배치
- 쓰기를 적절히 확장하기 위해 효과적인 기준으로 부하 분담
- 나눈다는 것은 관리의 차원에서 복잡성 증가
- 장애 포인트도 많아짐

업무영역별 분리



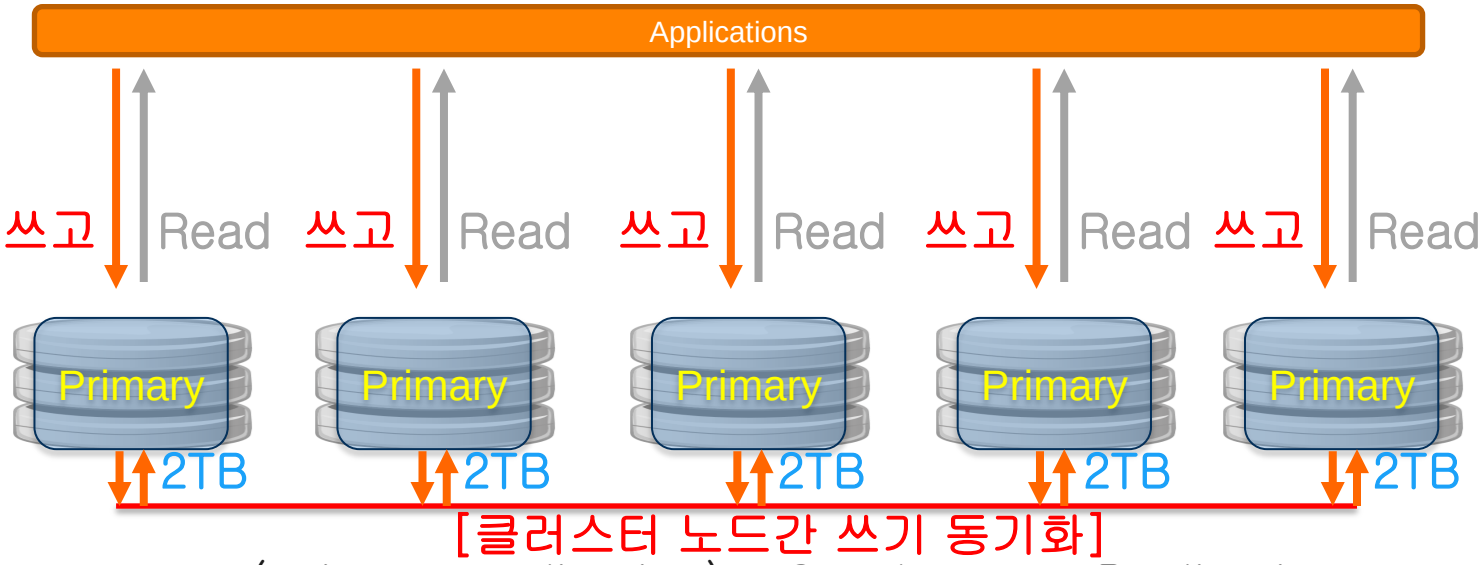
수백에서 수천대에 이르기까지 증가 …
잘 설계하고 나누기도, 잘 관리하기도 결코 쉽지 않다

C. Multi Master(R/W) 시도

Galera Cluster

MariaDB

Galera Cluster – High Availability

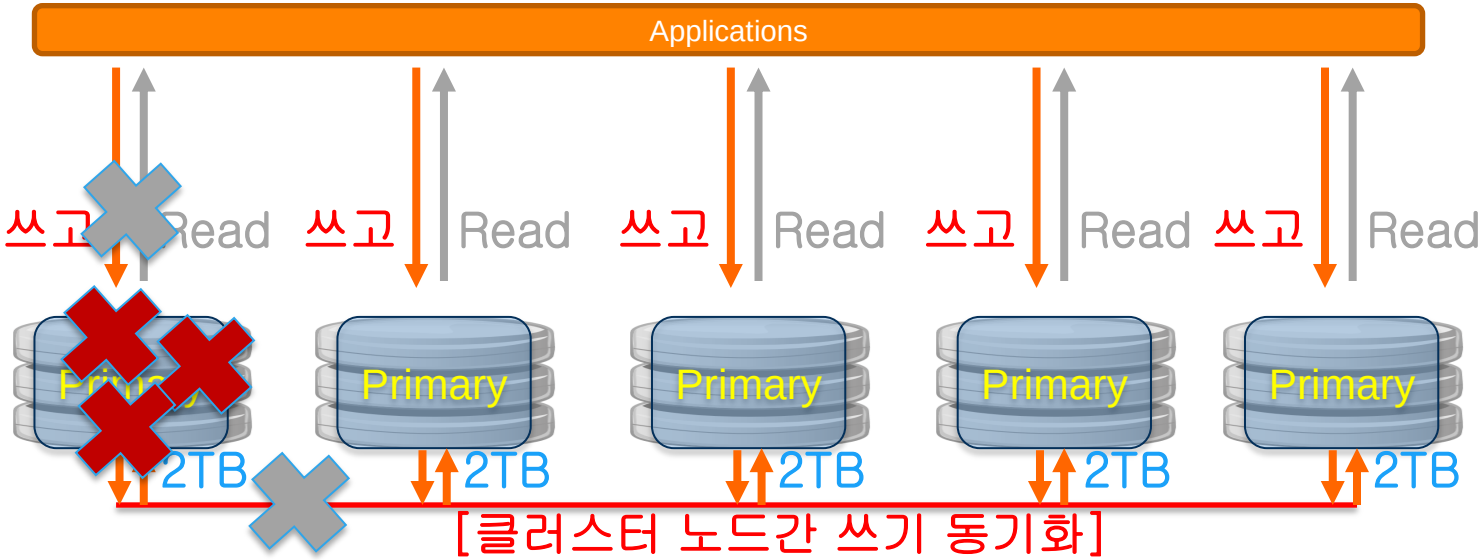


wsrep (write set replication) :: Synchronous Replication

However, in practice, synchronous database replication has traditionally been implemented via the so-called "2-phase commit" or distributed locking which proved to be very slow.

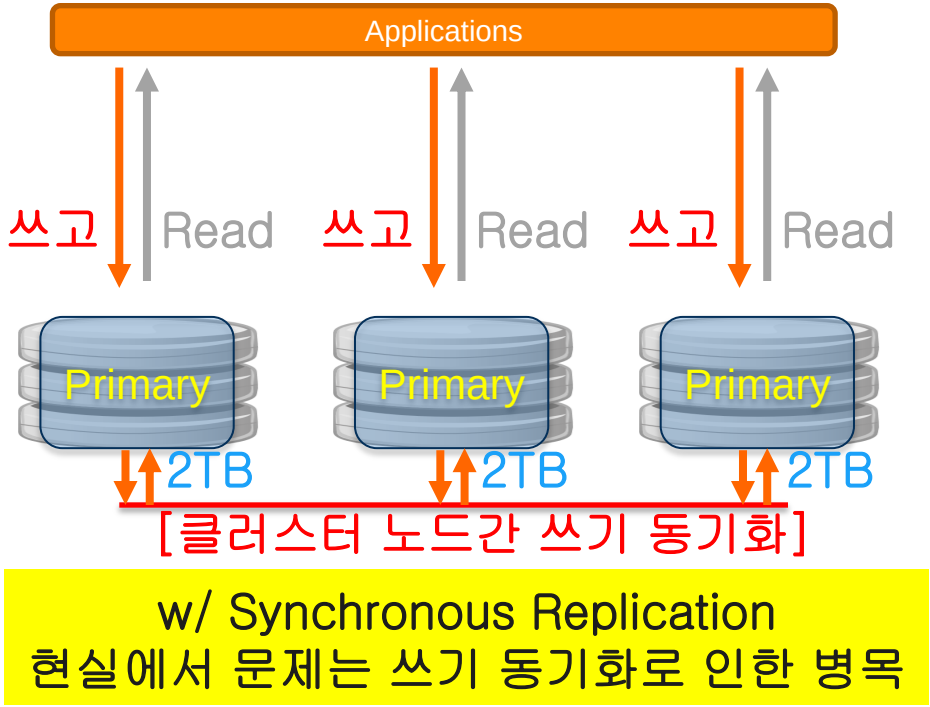
https://mariadb.com/kb/en/library/about-galera-replication/

Galera Cluster – High Availability



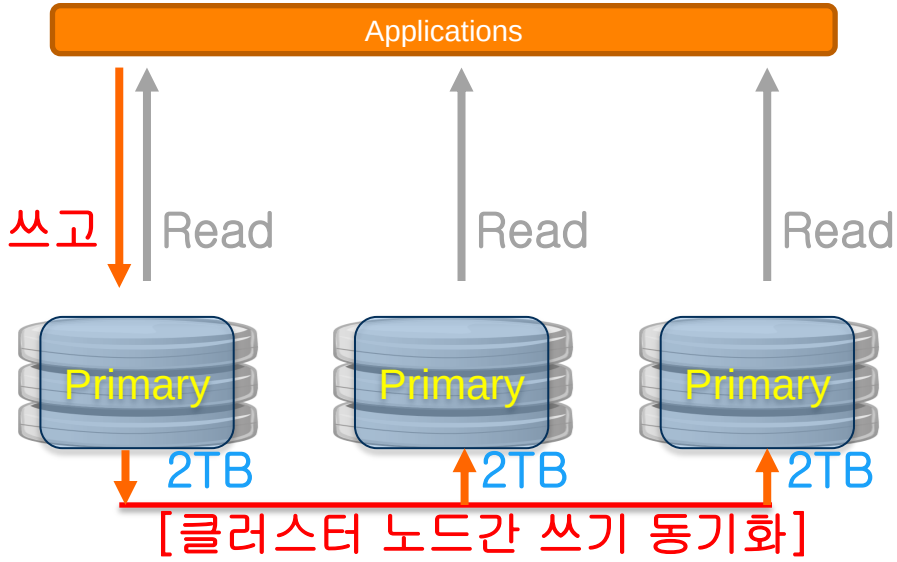
특정 노드 장애시, 이미 다른 노드들과 데이터 동기화 되어 있음 전제

Galera Cluster – High Availability



*HA : Node 장애시 지체 없이(데이터 정합성 확보) 다른 노드가 서비스를 처리

Galera Cluster – High Availability



Intensive Write 발생시 모든 노드마다 100% Write 반영해야함

Write 부하를 분담할 수 없으므로 Intensive Write Workloads 에는 적합하지 않음

현실은 오히려 쓰기 저하를 가져 옴
하지만 장애 상황에서 중단 없이 읽기 가능

일반적 사용1 : Write 가 매우 적고 Read 가 많으면서 HA 중요한 경우
일반적 사용2 : Read 가 절대 실패하면 안되는 요구사항

지금까지 본 기존의 방식은...

여전히 쓰기 확장의 한계
비동기 복제방식으로 인한 데이터 정합성 이슈

MariaDB

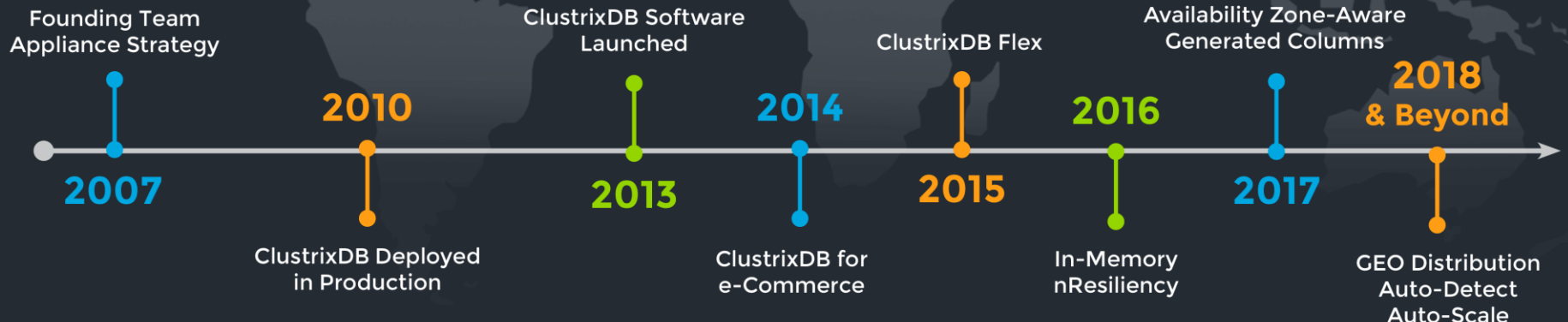
ClustrixDB

이제 진정한 Scale Out 을
만나보세요

MariaDB

ClustrixDB Product History

- Since 2007
- ★ - 13 Years
- Matured and reliable RDB



Why ClustrixDB ?

- Genius 한 자동 분산 데이터 아키텍처 및 핸들링
- 노드추가를 통해 읽기/쓰기가 원하는 만큼 선형확장됨
- 확장으로 인해 발생하는 급격한 관리비용 상승 없음
- Patent 된 Rebalancer 가 자동으로 관리
- 표준 SQL / MariaDB & MySQL Compatibility 지원
- 서비스 다운타임 최소화 실현

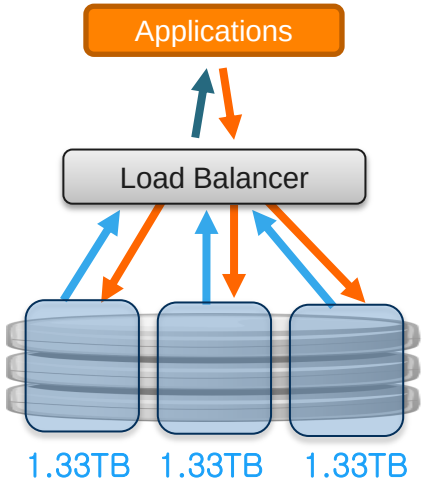
MariaDB

ClustrixDB

1. 데이터 분배

MariaDB

ClustrixDB 데이터 분배

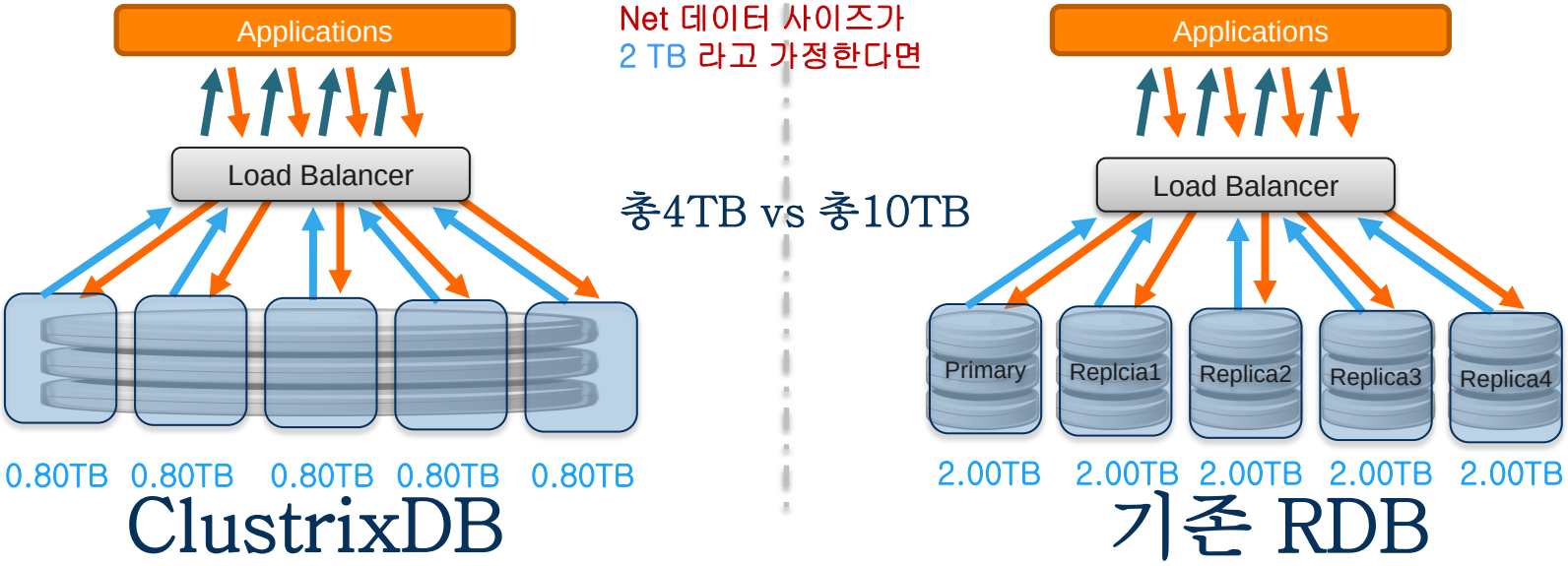


(2 TB * 2 Replicas) / Number of Nodes

3 node cluster => (4 TB / 3) = 1.33 TB
5 node cluster => (4 TB / 5) = 0.80 TB
9 node cluster => (4 TB / 9) = 0.45 TB

- 개별 노드에서 핸들링 해야 하는
  절대적인데이터량이 줄어 듬
- 개별 노드 입장에서 더 적은 resource 소요

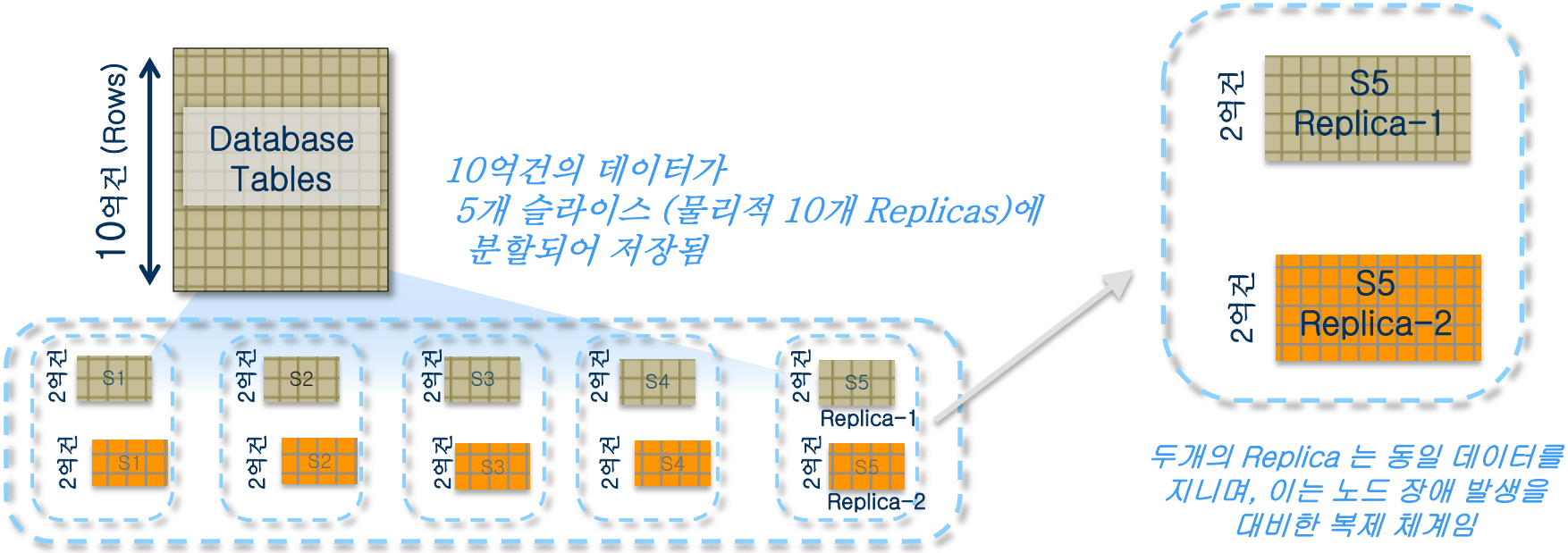
ClustrixDB Data Distribution



총 데이터 사이즈가 노드수 증가시 증가하지 않음
노드가 증가할 수록 한 노드가 처리해야할 일량이 줄어듬
노드별 핸들링할 데이터 량이 줄어듬으로 필요 Resource(cpu/memory/IO) 가 낮아짐

지능적 데이터 분배 – I

하나의 슬라이스는 두개의 리플리카로 구성

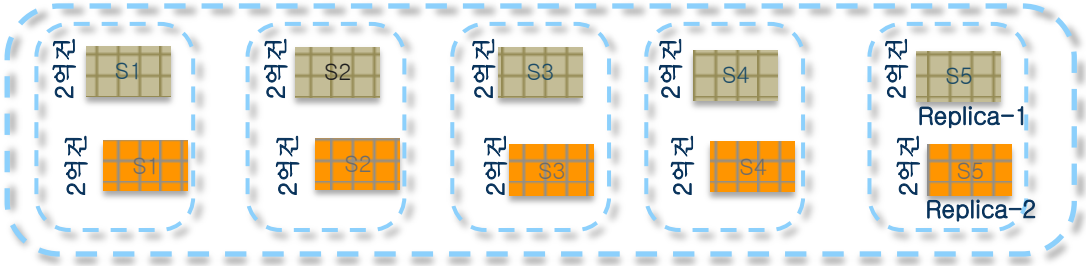


- Slice 개수를 500 개로 한다면 500 x 8GB = 4TB 를 수용함
- 데이터 사이즈가 늘어나면 자동으로 슬라이스 개수는 증가함

지능적 데이터 분배 – II

```
CREATE TABLE `sbtest1` (
`id` int(11) not null AUTO_INCREMENT,
`k` int(11) not null default '0',
`c` char(120) CHARACTER SET utf8,
`pad` char(60) CHARACTER SET utf8,
PRIMARY KEY (`id`) DISTRIBUTE=1
) REPLICAS=2 SLICES=5
```

Consistent Hash Function f(x)
= f(DISTRIBUTE=1)
= f(id)
= SLICE NO.



지능적 데이터 분배 – III

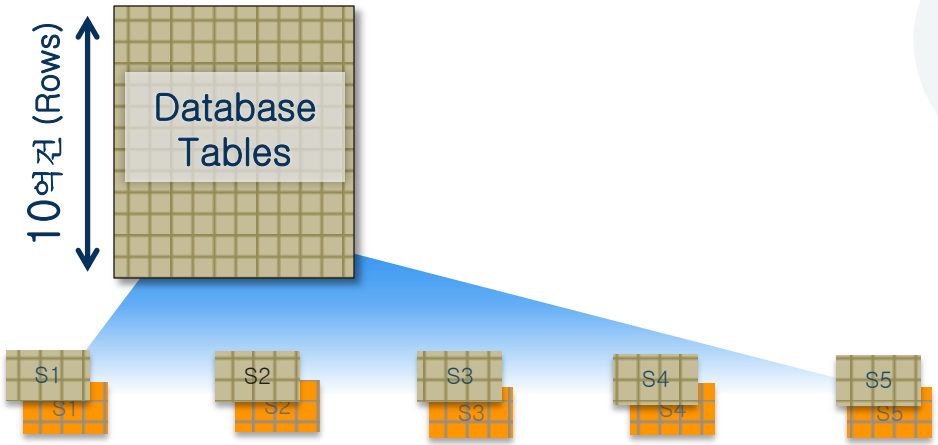
- 내부적으로 Consistent hashing (노드의 추가 및 제거를 수용하는) 을 사용하여 분배

```
MySQL [system]> select * from hash_distribution_map
                where hash_dist = 65017959193792286 74;
+---------------------------+---------------------------+---------------------------+
|        hash_dist          |           hash            |           slice           |
+---------------------------+---------------------------+---------------------------+
| 65017959193792286 74      | 3689348814741910323       | 65017959193835376 66      |
| 65017959193792286 74      | 7378697629483820646       | 65017959193835386 90      |
| 65017959193792286 74      | 11068046444225730969      | 65017959193835397 14      |
| 65017959193792286 74      | 14757395258967641292      | 65017959193835407 38      |
| 65017959193792286 74      | 18446744073709551615      | 65017959193835417 62      |
+---------------------------+---------------------------+---------------------------+
```

*SLICE NO 65017959193835376 66 은
기본적으로 두개 이상의 REPLICA 로 구성되며 각 REPLICA 는 서로 다른 노드에 저장됨

지능적 데이터 분배 – IV

- 테이블을 슬라이스들로 자동 분할
- 모든 슬라이스는 다른 노드에 복제본을 지님
  - 슬라이스는 자동분배 및 자가복원됨



ClustrixDB

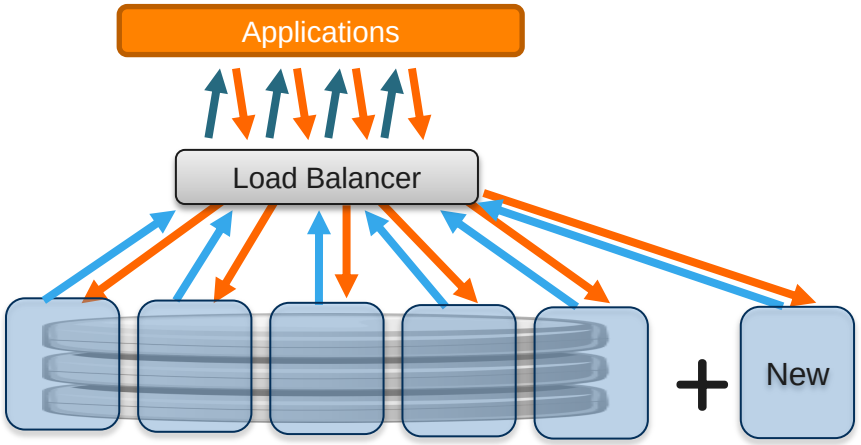
예) 10억건의 테이블은 각 2억건의 5개 슬라이스로 구성됨
하나의 슬라이스는 장애를 대비한 복제본을 가짐

ClustrixDB

2. 노드 확장

MariaDB

ClustrixDB 의 확장성



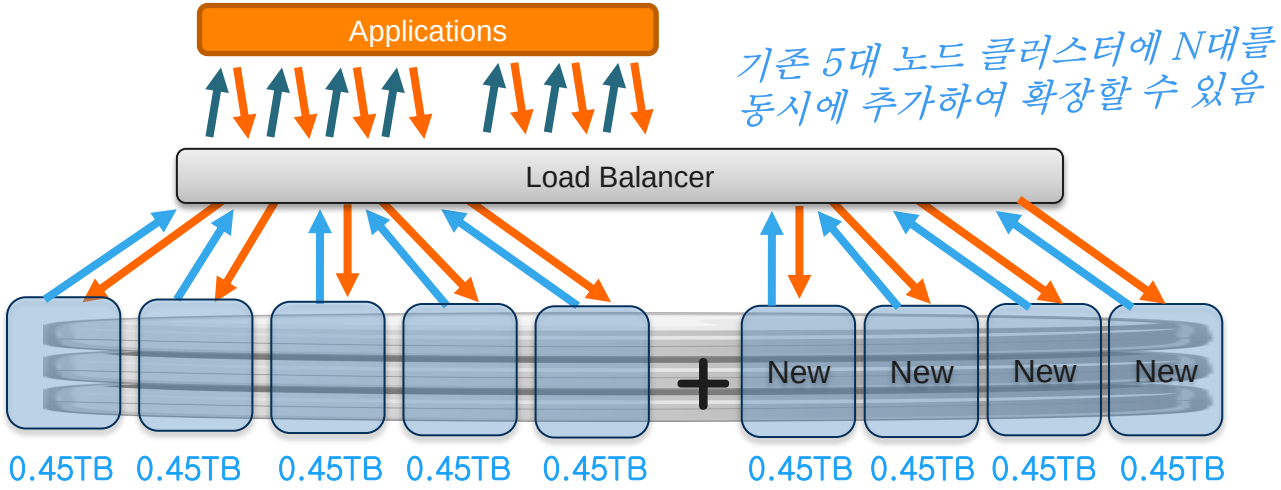
논리적으로 여전히 하나의 DB 이면서 동시에 매우 손쉬운 노드 연결 및 확장

1. 노드 추가 전 : 0.80TB 0.80TB 0.80TB 0.80TB 0.80TB

2. Copying and Moving Data Slice – Container Level (8G) :
현 운영 서비스에 영향을 최소화하며 Smooth 하게 알아서 자동으로 진행 (By Rebalancer patented)

3. 노드 추가 후 : 0.67TB 0.67TB 0.67TB 0.67TB 0.67TB 0.67TB

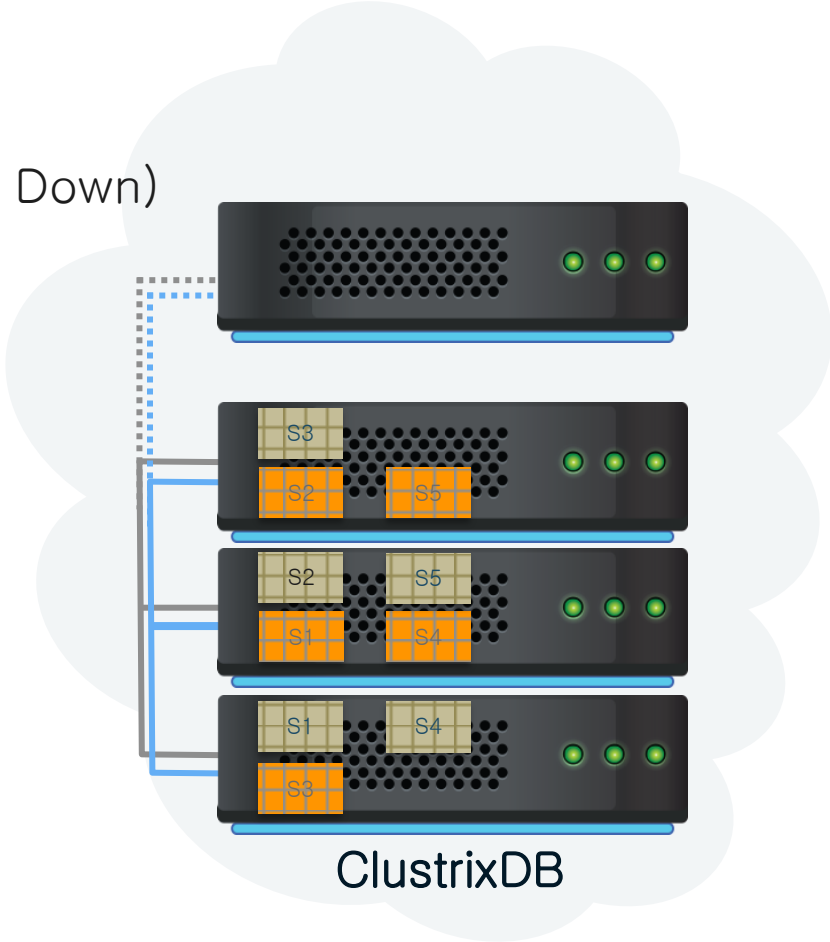
ClustrixDB 의 확장성



기존 5대 노드 클러스터에 N대를 동시에 추가하여 확장할 수 있음

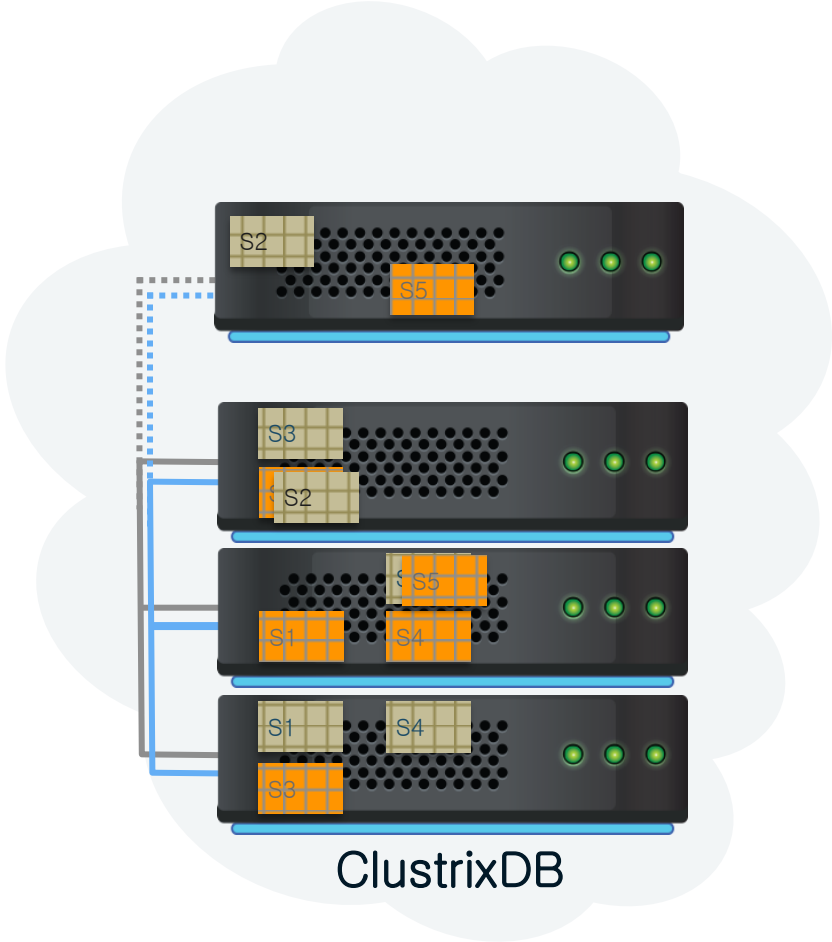
노드 추가

- 쉽고 간단한 노드 추가 및 제거(Flex Up & Down)
  - 서비스 중단 최소화
- 데이터는 자동으로 균형 분배 및 조정
  - 노드 균형 분배 중에라도 테이블은 읽고/쓰기 가능한 온라인 상태 유지
- 모든 노드에서 writes + reads 처리
  - Flex Up 이후 자동으로 데이터 분배 (1/N) 및 작업부하를 모든 노드에 균등하게 분산



노드 제거

- ClustrixDB 는 노드손실을 자동 감지
  - 시스템 자동 재보호(손실 슬라이스 복구)
  - 잔여 노드들에 데이터 자동으로 균등 재분배
- 손실된 노드의 데이터 신속 재 보호
  - 데이터 재 보호 중에도 테이블의 읽고 쓰기는 여전히 가능
- 자동으로 자가 복원
  - 클러스터가 자동 자가복원 후에 완벽하게 보호되어 작동함



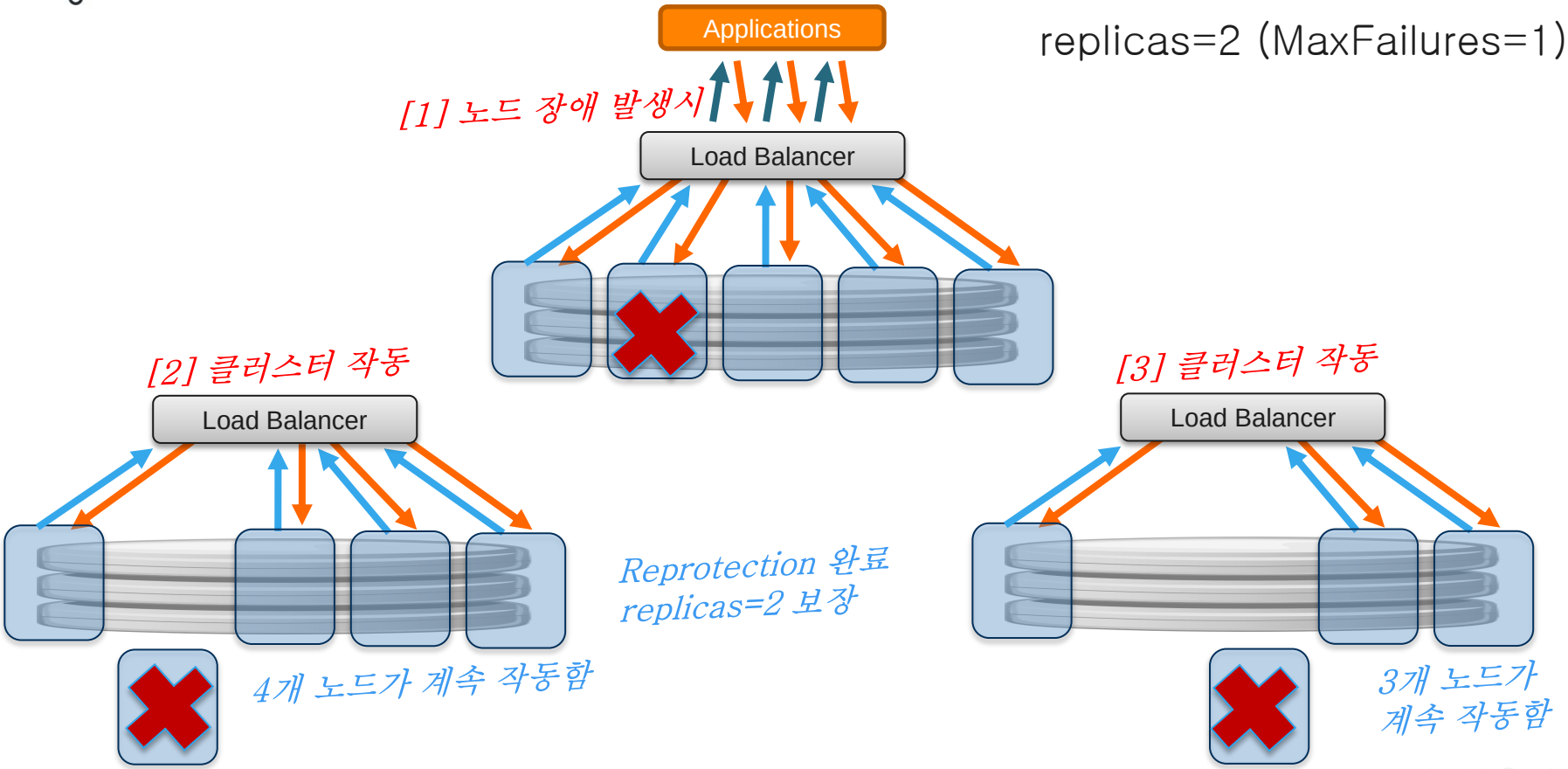
ClustrixDB

3. n-Resiliency & Rebalancer

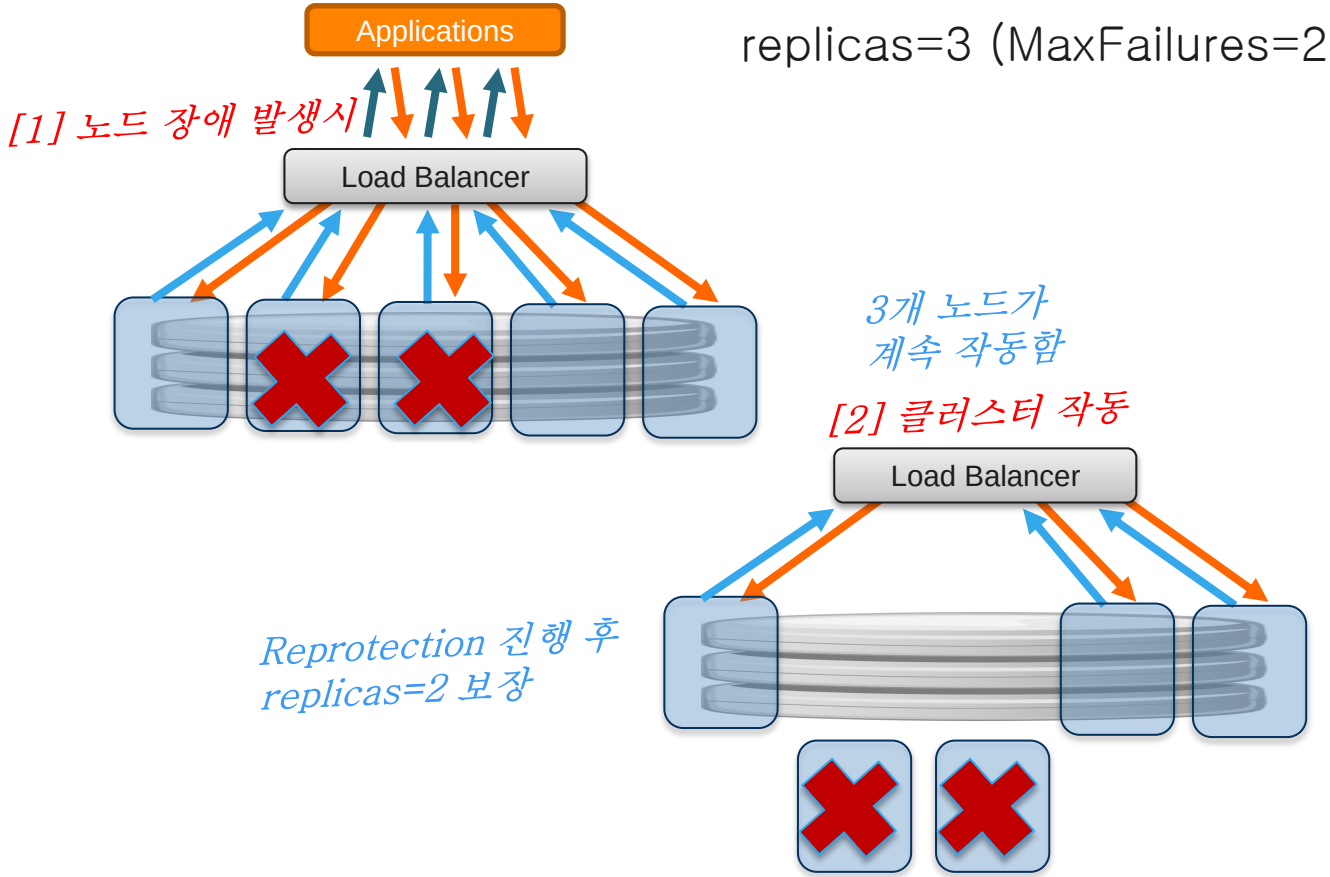
replicas=3 (MaxFailures=2)
“동시에” 2대의 노드장애가 발생하더라도
나머지 한대가 있기에 서비스는 계속됨

MariaDB

노드장애 자동복구 예시



노드장애 자동복구 예시


Applications
[1] 노드 장애 발생시
Load Balancer
replicas=3 (MaxFailures=2)
3개 노드가
계속 작동함
[2] 클러스터 작동
Load Balancer
Reprotection 진행 후
replicas=2 보장

리밸랜서(Rebalancer)

Q: 클러스터 환경에서 데이터가 잘 분산되도록 하려면 어떻게 해야 합니까 ?

A: Clustrix Rebalancer 가 알아서 처리 하도록 두면 됩니다

리밸랜서(Rebalancer)가 자동으로:

- **초기 데이터 분배:** 모든 노드들에 균등하게 슬라이스(조각 데이터)를 분배
- **데이터 폭증 부하 분산:** 특정 슬라이스 커질시 작은 슬라이스로 분할 분배
- **Flex-Up/Flex-Down:** 노드 추가 삭제에 따라 슬라이스 재분배 및 복구
- **노드 다운 장애 복원:** 다운된 노드의 슬라이스 복제를 통한 자가 복원
- **데이터 쏠림 현상(Skewed Data)해소:** 쏠림 슬라이스를 데이터 노드에 균등하게 재분할 분배
- **핫 슬라이스 분할 분산(Hot Slice Balancing):** 빈번한 요청이 있는 슬라이스를 찾아 노드간 재 분산

…DB 가 읽기&쓰기 작업 동안에도 리밸런싱 수행 가능

ClustrixDB

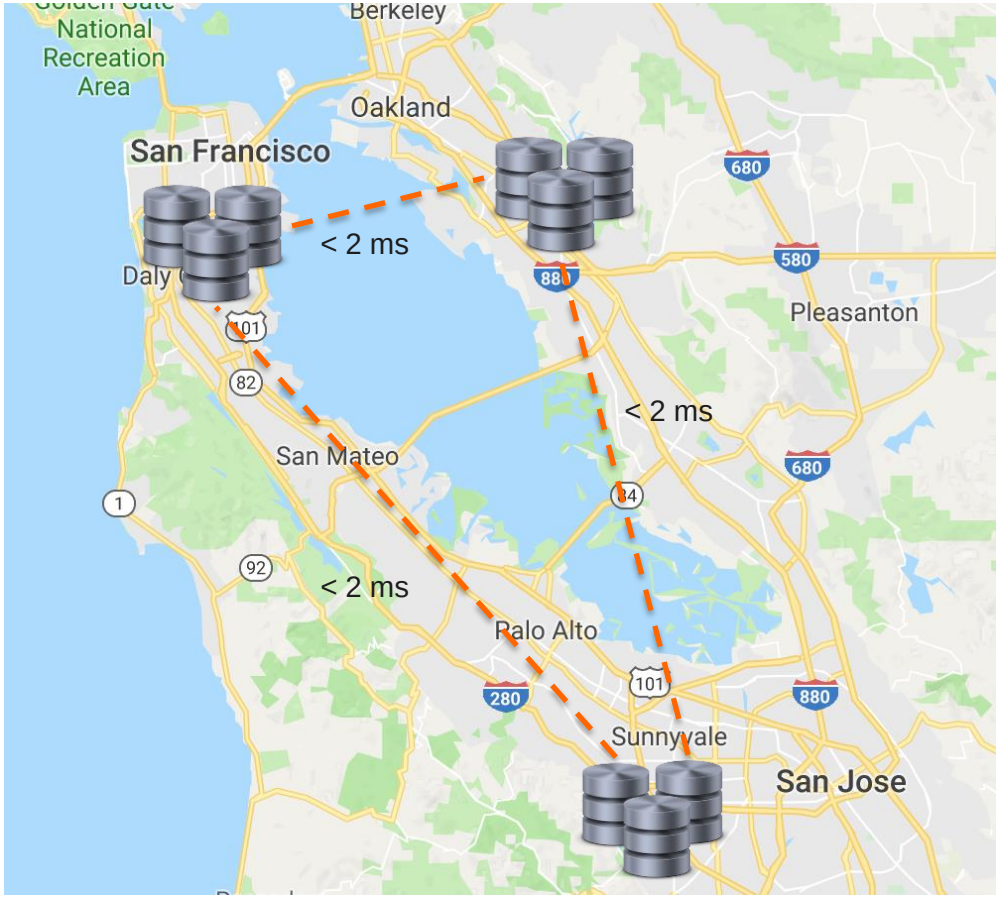
4. Multi-Zone Availability Cluster
데이터 센터 전체 장애 감내

MariaDB

Metro Clusters – Multi-Zone Availability

New in ClustrixDB 9!

- *n*Resliency 위에 구축
  - 하나의 데이터 센터 전체 장애 발생에도 데이터 보호
- ClustrixDB 9의 노드는 여러 지역에 걸쳐 "확장"될 수 있음
  - 메트로 지역
  - 또는 AWS Region (즉, 미국 동부)
- 복잡한 복제 구축 설정 또는 유지관리 불필요
- 모든 노드는 100% ACID 준수
  - 특정지역(zone) 장애가 발생한 경우에도 완전한 일관성 보장



ClustrixDB

5. 기타

MariaDB

> 몇가지 더 ...

- clustrix_import 데이터 임포트 병렬처리
- sysbench 표준 테이블(PK 있음) 1억건 INSERT 테스트시
  - 3노드(8core) 클러스터에서 12분 8초
  - 10노드(8core) 클러스터에서 3분 45초
- hash_aggregate_partial and combine 실행계획
  Aggregation 병렬 처리
  Node parallelism x Core level parallelism

> 새로이 출현하는 솔루션 그리고
특히 오픈소스 데이터베이스는 **매우 잘 알고 써야 제대로 씁니다**

- 모든 프로덕트는 실제 정말 잘 쓰려면 아키텍트에 맞는 설계
- 기본설계요소 : PK, DISTRIBUTE KEY, SLICE, REPLICAS
- 강점을 살리는 사용 그리고 약점을 회피하는 사용
- 쓰기에 강하고 분산 시스템 원천적 특징상 네트워크 부하
  ex. Covered Index
  especially in case Index Range Scan & Table Access
- 다시 한번 더 네트워크
  ex. ALLNODES, FK

Deep Dive 를 위한 공식 문서

ClustrixDB Online Document

온라인문서
http://docs.clustrix.com/display/CLXDOC/Distributed+Database+Architecture
http://docs.clustrix.com/display/CLXDOC/Data+Distribution

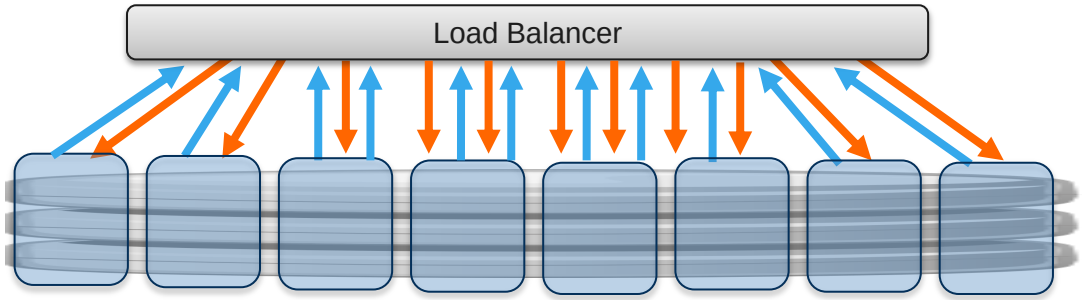
기술자료
https://support.clustrix.com/hc/ko/categories/202038843-기술자료

ClustrixDB

실 적용 사례

MariaDB

적용 I – 거래소
Scale Out + Spike 핸들링 + 무중단서비스



- ✓ Easy Scale Out
- ✓ Peak Time Handling
- ✓ 급등 급락 Event Spike
- ✓ Marketing Campaign

On Premise and AWS (Optimized AMI 제공)

적용 II – Intensive Write 멀티 AZ 클러스터

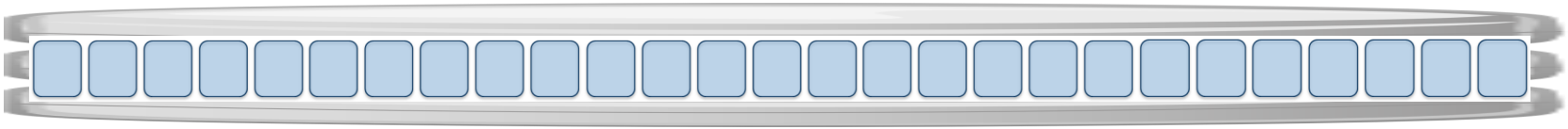
- ✓ 한 AZ 전체 장애 대응
- ✓ Update Intensive over 50% workloads
- ✓ Peak Time Handling

적용 III – Game

- ✓ 빠르며 쉬운 확장
- ✓ 갑작스런 서비스 증가에 따른 대응
- ✓ 게임서비스 정상적 운영

적용 IV – 초대용량 데이터 핸들링

✓27 대 노드 클러스터



Thank you

facebook.com/groups/mariadbkorea

2019.09.25

조현기 (Jacob Jo) jacob.jo@mariadb.com
SQLP / DAP / PMP