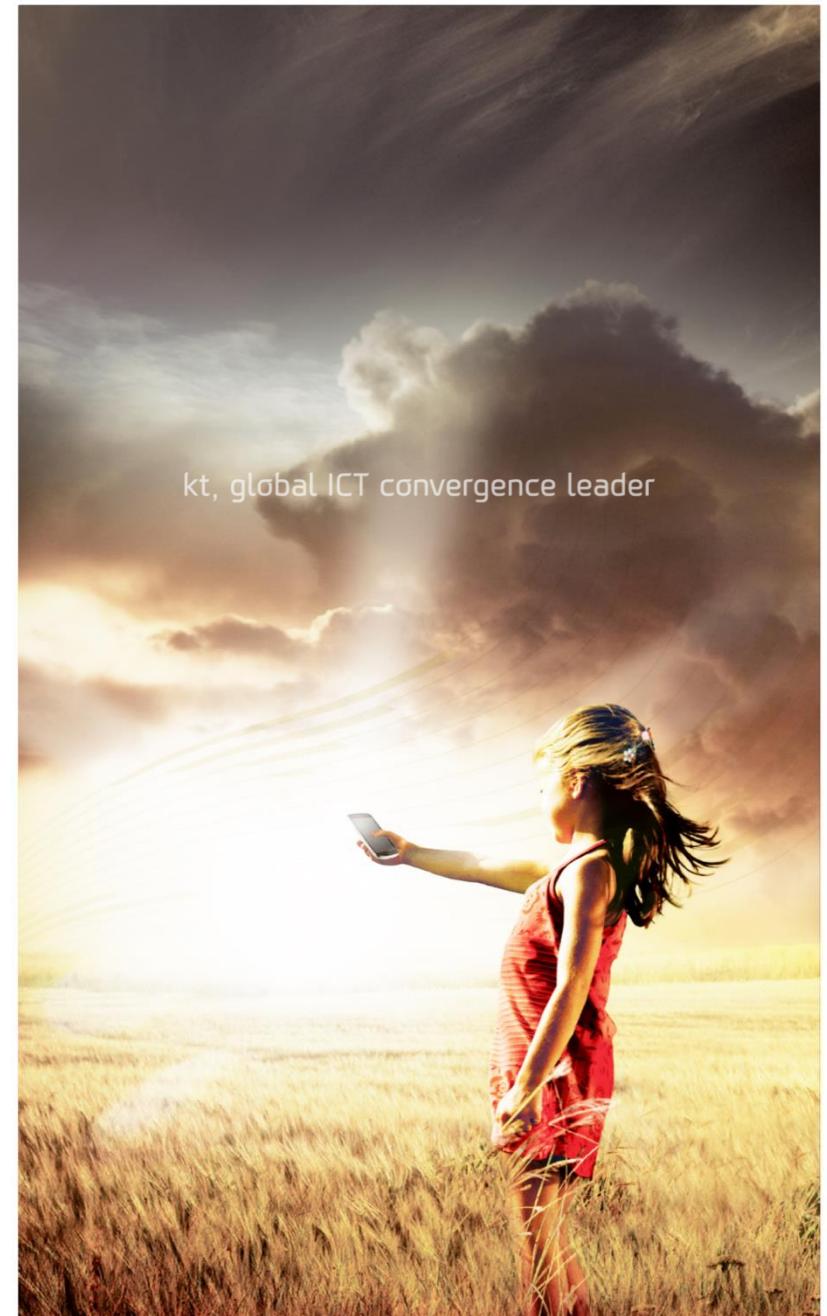


이중화, DR 구축 on ucloud



1 이중화, DR의 개념

2 ucloud 인프라의 redundancy 구성

3 Redundancy가 제공되는 서비스

4 VM의 이중화, DR 구성

5 DB 이중화, DR

01 이중화, DR의 개념

이중화

스패어 타이어 (Rack)
영향 제한적(VM or Server)
Less costly, same place

- 서버나 VM의 장애시 동일한 서버나 VM을 예비로 준비하여 무중단 서비스를 제공하기 위한 시스템 구성 방법



DR(Disaster Recovery)

자동차 (IT Infra)
영향 광범위(Data Center)
Costly, Different places

- 자연재해나 설비로 인한 데이터 센터 장애시 또 다른 데이터 센터에 구축된 IT 인프라를 통해 무중단 서비스를 제공하는 방법



&

1 이중화, DR의 개념

2 ucloud 인프라의 redundancy 구성

3 Redundancy가 제공되는 서비스

4 VM의 이중화, DR 구성

5 DB 이중화, DR

01 ucloud 인프라의 redundancy 구성 - 데이터 센터 이중화

천안과 목동을 DR 구성하여 데이터 센터 이중화 구성
김해GDC도 활용 가능

천안 CDC

Main Cloud
Data Center



목동 CDC

최초 Cloud 시스템
수용



김해 GDC

Global Data Center



&

&

02 ucloud 인프라의 redundancy 구성 - 전력 설비 이중화

변전소 이중화

- 전원 공급원인 변전소를 서로 다른 2개를 이용하여 하나의 변전소에 장애나 자연재해가 발생하더라도 전원이 공급될 수 있도록 함



UPS N+1 with 발전기

- 무중단 전원 공급장치(Uninterruptible Power Supply)를 N+1 로 가져가고 UPS에 대한 백업으로 디젤 발전기를 준비하여 7일간 외부전원 공급없이도 데이터 센터 운영 가능



분전반 이중화

- 분전반 2개를 랙에 있는 2개의 power strip에 각각 공급하여 이중화 구성



랙 케이블 Power strip 이중화

- 랙으로 들어오는 인입선을 2개로 이중화하고 랙 내 power strip도 이중화 구성됨



03 ucloud 인프라의 redundancy 구성

 가용성 Zone(AZ)	인프라가 천안, 목동, 김해 등으로 분산되어 있어 물리적인 DR을 제공함	 
 서버	WAS나 Web을 active-active 혹은 active-standby 구조로 분산하여 이중화 구성 가능	
 스토리지	Block storage형태의 데이터 볼륨을 스냅샷을 통해 저장하고 필요한 때 마운트하여 사용 가능	
 DB	스냅샷 기반의 백업 제공. MS-SQL cluster, replication 및 Oracle RAC 구성 가능	
 네트워크	VR(가상라우터)를 이용하여 VR LB 제공 고성능 LB 제공 가능	

04 ucloud 인프라의 redundancy 구성



서버 이중화

- 서버의 NIC카드 이중화와 내부 Disk 이중화, PSU 이중화
- HA 기능으로 서버 failure 시에 VM들은 다른 서버에서 auto-restart
- 서버 유지보수 작업 시에는 다른 서버로 무중단 마이그레이션



네트워크 이중화

- 모든 네트워크 장비들의 이중화를 통한 Single Point of Failure 제거
- 서비스 네트워크와 각 시스템을 관리하는 관리망 분리
- Service Network와 Storage Network를 분리하고 각각 이중화



스토리지 이중화

- Storage controller 이중화 구축
- RAID 적용



1 이중화, DR의 개념

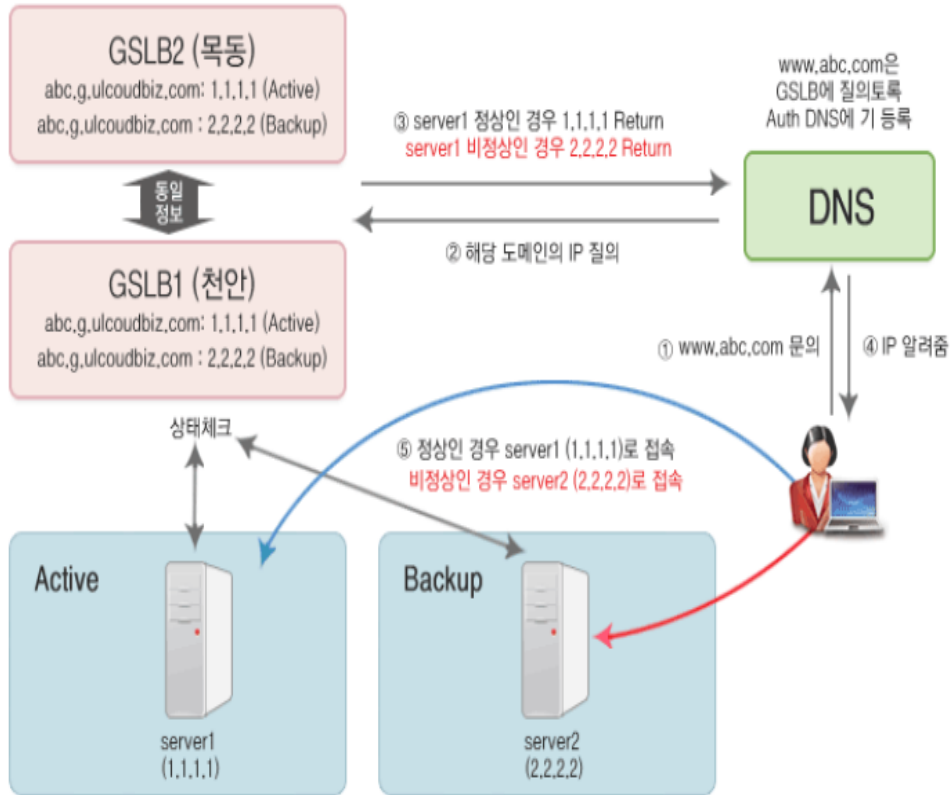
2 ucloud 인프라의 redundancy 구성

3 Redundancy가 제공되는 서비스

4 VM의 이중화, DR 구성

5 DB 이중화, DR

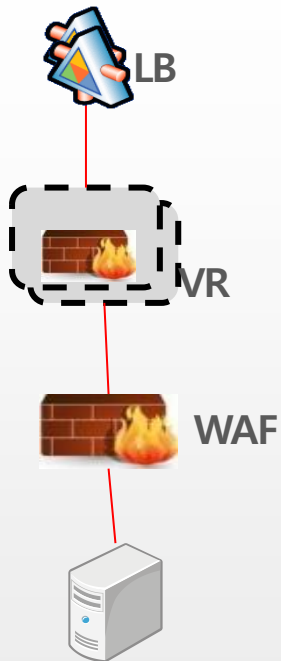
01 Redundancy가 제공되는 서비스 - GSLB



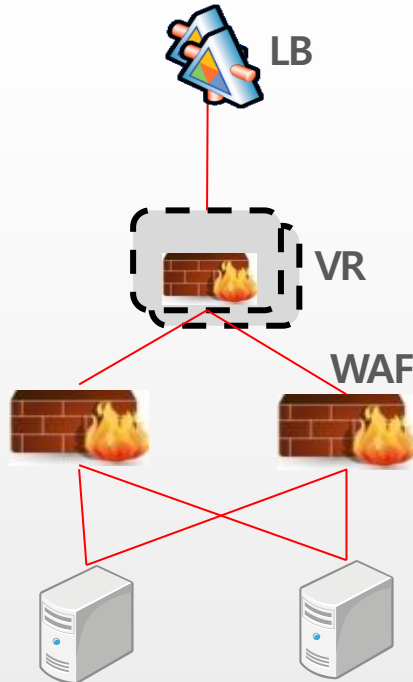
- ✓ **용도**
 - 데이터 센터간의 물리적 이중성 확보
- ✓ **도메인 질의**
 - 해당 도메인을 GSLB에 질의하도록 Auth DNS에 기 등록해야 함
- ✓ **IP Return**
 - GSLB는 Active or backup server중 정상 운용 중인 서버 IP 리턴
- ✓ **서버 접속**
 - 정상 운용 중인 서버에 접속함

02 Redundancy가 제공되는 서비스 -LB, Firewall, WAF

Single mode WAF



Dual mode WAF



LB, FW, WAF 이중화

LB
이중화

- LB(Load balancer)는 안정적인 서비스를 제공하기 위해 default 이중화 구성 (종량제로 변경 예정)

Firewall(VR)
이중화

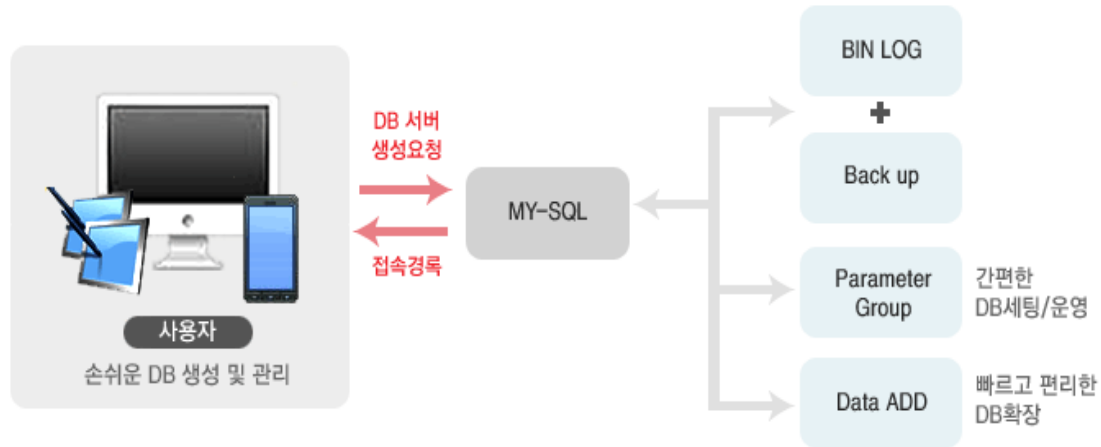
- VR은 가상 서버를 생성하는 순간 Default로 이중화 구성된다

¹⁾WAF
이중화

- WAF를 생성할 때 하나만 생성하면 1대의 가상서버에만 적용할 수 있으므로 서버가 2대 이상이라면 반드시 2개의 WAF를 구성하도록 한다

03 Redundancy가 제공되는 서비스 - ucloud DB

MySQL replication 기능을 활용한 ucloud DB 이중화



이중화 원리

- MySQL Replication을 이용하여 Master - Slave 이중화
- Linux CentOS 기반

특징

- 사용자가 설정한 복구 가능 기간 만큼 자동 backup 수행
- Snapshot 기능을 통한 백업
- DB에 대한 자동 패치 기능 제공
- 모니터링 기능 제공
- Multi Availability zone 제공(물리적, 지리적 DR 기능 제공)

제공 기능

- MySQL Replication의 slave를 read용으로 scale out 기능 제공
- Replica(slave)그룹 관리 제어 기능
- MySQL multi version 지원
- DB 접근 제어 그룹 관리와 ucloud DB인증서 발급 및 관리 기능 등의 제공으로 보안 강화
- DB 기능의 Open API 기능 제공

1 이중화, DR의 개념

2 ucloud 인프라의 redundancy 구성

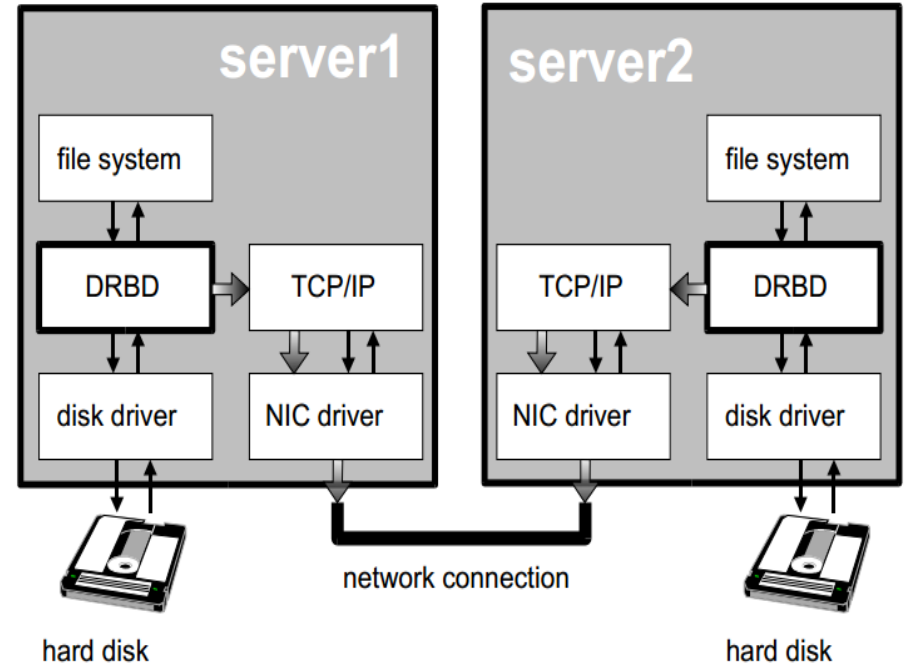
3 Redundancy가 제공되는 서비스

4 VM의 이중화, DR 구성

5 DB 이중화, DR

01 VM의 이중화, DR 구성 - DRBD (1/2)

DRBD	<p>Distributed Replication Block Device</p> <ul style="list-style-type: none"> • 블록 디바이스(하드디스크, 볼륨)에 적용 • Active-Standby (Master-Slave 구조) • Asynchronous, synchronous 구성 모두 가능 • Network based RAID1(Mirroring)
Application	<p>Base application</p> <ul style="list-style-type: none"> • a conventional file system • Shared disk file system • Logical block device(LVM) • Block device를 사용하는 모든 어플리케이션
Advantage	<p>Advantage</p> <ul style="list-style-type: none"> • Real time update on slave • Consistency guaranteed by DRBD • Fast recovery after failover
Dis-advantage	<p>Disadvantage</p> <ul style="list-style-type: none"> • Duplicate data • Replication 하는 동안 slave DRBD device를 사용할 수 없음 • write performance is reduced



파일 시스템이 DRBD를 통해 데이터를 disk driver 전달하고 disk driver가 물리적인 disk에 기록한다. 이 때 DRBD는 TCP/IP 통신을 통해서 클러스터로 구성된 서버2의 DRBD에 알리고 이를 통해서 데이터는 mirroring 된다.

01 VM의 이중화, DR 구성 - DRBD (2/2)

Protocol A

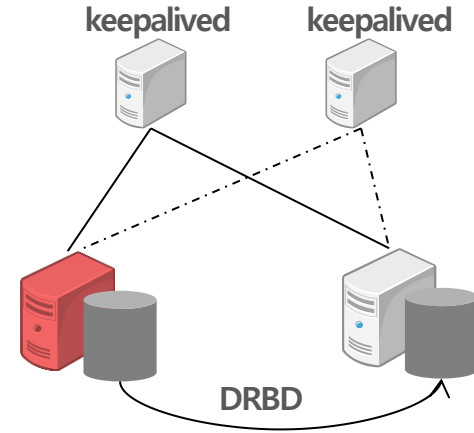
Write IO는 쓰기가 로컬디스크와 로컬 TCP IP send buffer에 도달하면 완료된 것으로 간주한다

Protocol B

Write IO는 쓰기가 로컬 디스크와 remote buffer cache에 도달하면 완료된 것으로 간주한다

Protocol C

Write IO는 쓰기가 로컬 디스크와 remote host의 로컬 디스크에 도달하면 완료된 것으로 간주한다.



<auto failover를 위한 구성>

DRBD failover

- Filesystem check 가동(5초)
- mysqld is started(1~5초)
- InnoDB crash recovery(1분~몇 시간)
- 서버가 가동됨
- **Auto-failover**를 가능하게 하기 위해 **keepalived**를 이용하여 운용 중인 서버의 장애 파악 (**keepalived 이중화**)
- 두 노드의 데이터 Consistency를 우선적으로 고려 할 때 적합한 구성이고 다른 솔루션과 결합하여 최대한의 uptime(HA)를 위한 구성도 할 수 있음

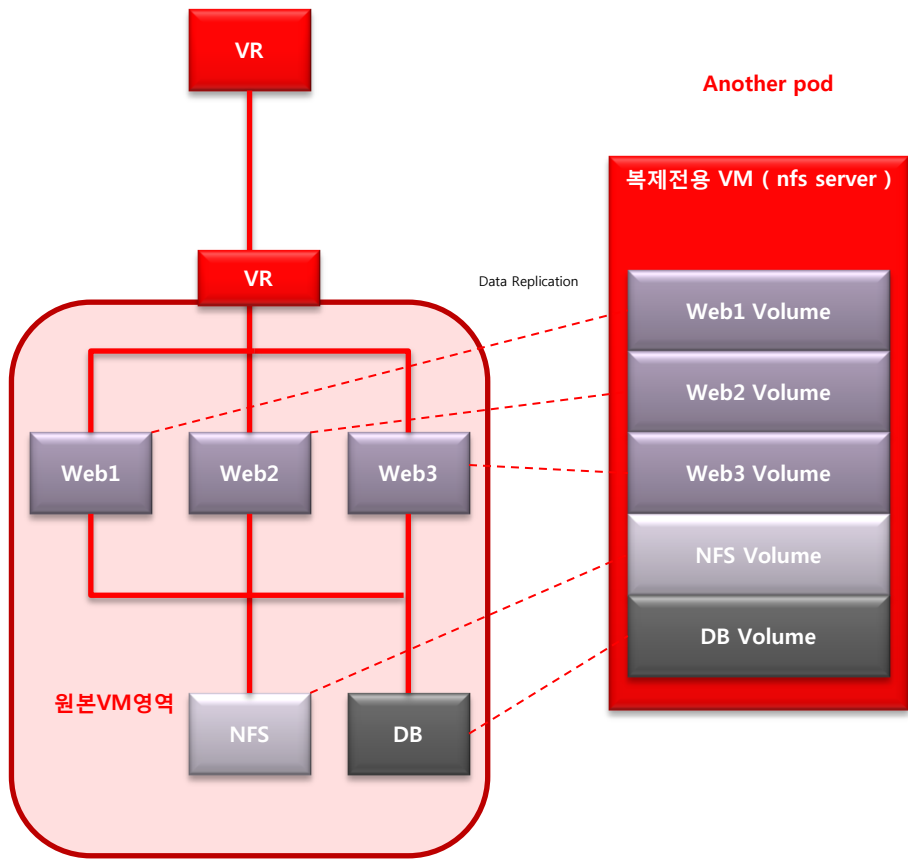
VM의 이중화, DR 구성 - Steeleye (1/2)

※ ucloud에서 제공하지 않는 유료솔루션 (별도 구매 필요)

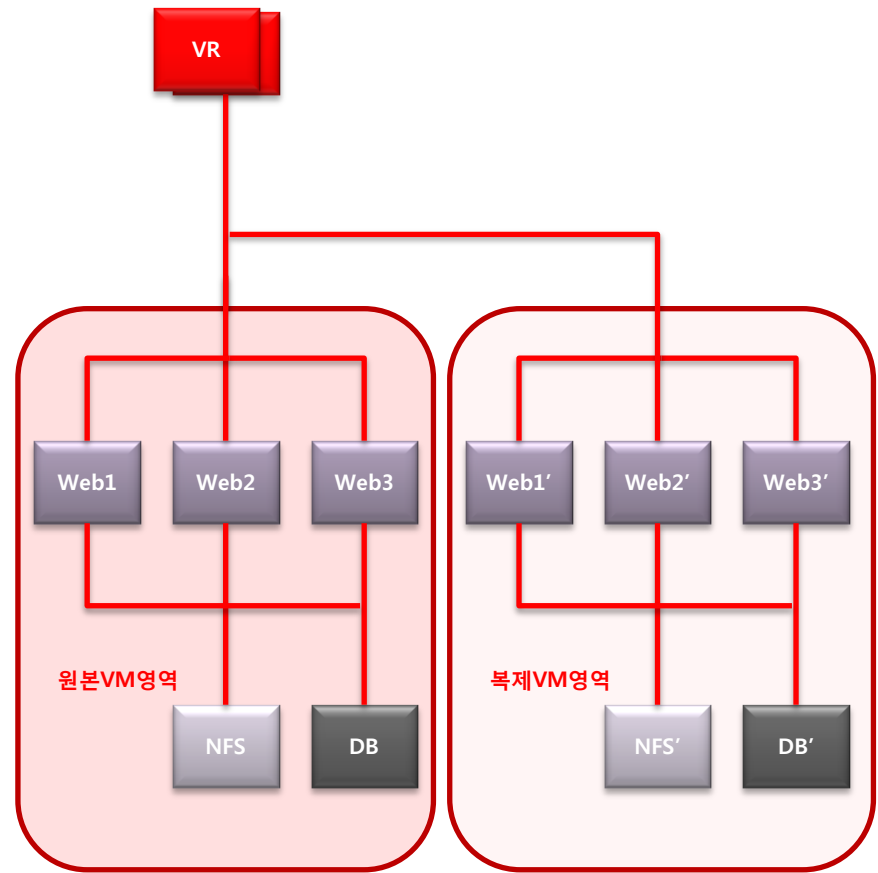
LifeKeeper DataKeeper	LifeKeeper <ul style="list-style-type: none"> Linux/Windows 서버용 HA솔루션 DataKeeper <ul style="list-style-type: none"> Block 단위 또는 file system단위 Data replication 솔루션 (Synchronous, Asynchronous 지원) 	LifeKeeper DataKeeper in ucloud	<ul style="list-style-type: none"> 복제 및 HA 서비스용으로 사용할 VM에 복제전용 POD에 VM 생성후 데이터 복제 원본 VM에 데이터 장애시 복제 VM을 이용하여 서비스 가능 복제할 VM의 자원은 VM, Storage 각각 2배 소요
Application	Data replication only <ul style="list-style-type: none"> 데이터에 대한 복제만 제공 VM HA + Data replication <ul style="list-style-type: none"> VM의 HA 및 데이터의 복제 서비스 제공 	서버 구성 in ucloud	<ul style="list-style-type: none"> Web2,Web2' = Windows 2008 64bit template + 200GB Data Disk Web1,Web1',Web3,Web3' = Windows 2003 32bit template + 200GB Data Disk NFS,NFS',DB,DB' = CentOS 5.4 64bit template + 200GB Data Disk
Advantage	Advantage <ul style="list-style-type: none"> 서버의 장애시 HA 제공 VM의 복제 서비스도 제공 VM 및 서버 실시간 백업 제공 CPU, 메모리 등 복제시 자원 소모가 적음 	Failover + Volume up	<ul style="list-style-type: none"> Web server = 100 sec(avg) NFS 서버 = 60 sec(avg) DB server = 40 sec(avg)
Dis- advantage	Disadvantage <ul style="list-style-type: none"> 추가 자원 필요(서버 혹은 VM 또는 스토리지) 고가의 라이선스 비용 	복제 성능 및 부하	<ul style="list-style-type: none"> Initial replication 속도: 40MB/s 100GB volume : 50 min, 1TB volume : 8.5 hr 초기 복제시 CPU 사용율은 1vcore VM 기준 평균 10% 상시 synch시 CPU 사용률은 2~5% 초기 데이터 복제시에도 서버 서비스 가능

VM의 이중화, DR 구성 - Steeleye (2/2)

Data replication only



HA + Data replication



03 VM의 이중화, DR 구성 - RSync

Rsync

File synchronization utility SW

- Data transfer를 최소화하면서 파일과 디렉토리를 원격 복제하는 오픈 소스 솔루션으로 윈도우용은 delta copy라는 솔루션이 있다.

How it works

- File 단위의 복제 방식으로 동기화 방식
- 수정시간 및 파일의 크기 변동을 체크하여 동기화할 파일을 결정한다.

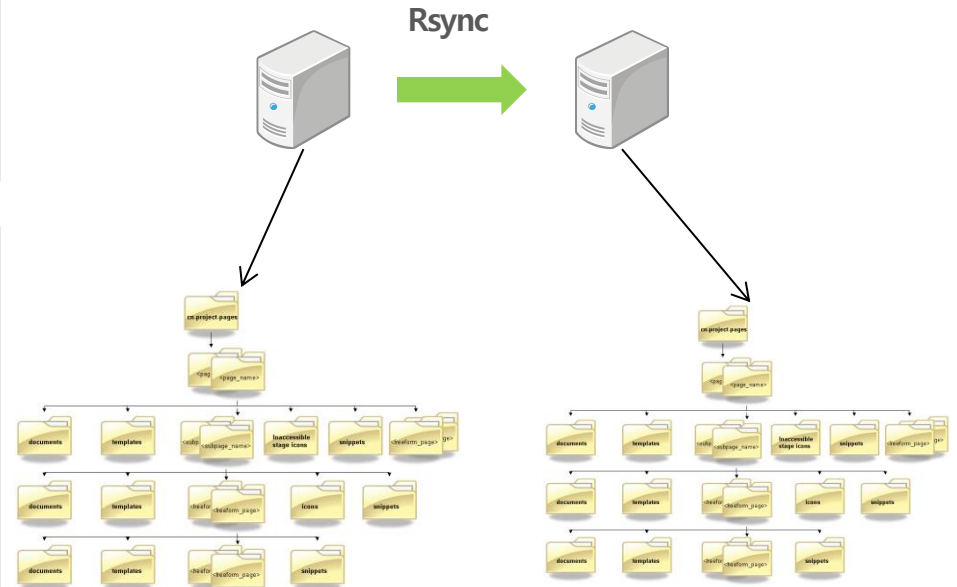
Advantages

- 라이선스 비용 없음
- 간단하고 효율적인 copy 및 move 지원

Disadvantages

- 실시간 동기화로는 사용이 어려움 (성능 문제)
- 동기화 파일 수가 많아지면 동기화 속도가 많이 느림
- 트랜잭션이나 변경 사항이 많은 서버에 대해서는 실시간 동기화가 거의 불가능
- log backup 등 용도가 제한적

How it works



※ 기타, deltacopy (windows용) 등 유사한 기능을 제공하는 다양한 솔루션이 있음

1 이중화, DR의 개념

2 ucloud 인프라의 redundancy 구성

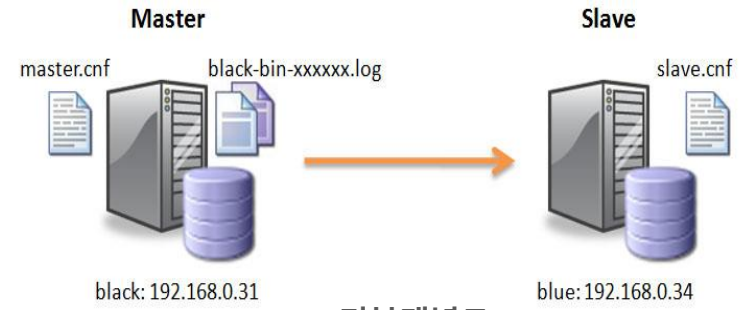
3 Redundancy가 제공되는 서비스

4 VM의 이중화 DR 구성

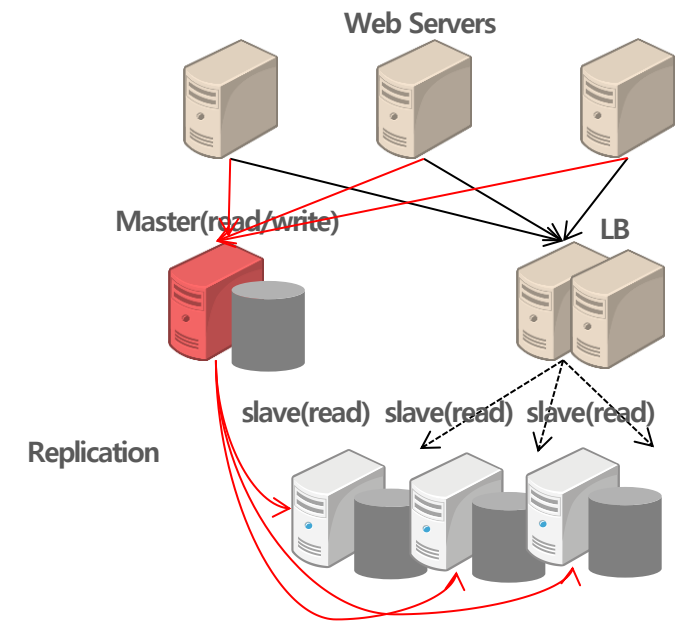
5 DB 이중화, DR

01 DB 이중화, DR - MySQL Replication (1/5)

<p>구성요소</p>	<ul style="list-style-type: none"> • 1 Master & possibly multiple slaves
<p>이중화 원리</p>	<ul style="list-style-type: none"> • Replication은 Slave Server가 replication을 위한 Request를 보낸다. • Master Server는 Slave Server의 Credential을 확인한 후 transaction file인 binlog파일의 event를 Slave Server로 보낸다. • Slave는 위 event를 바탕으로 relaylog를 쓴다. • Slave의 thread가 relaylog를 읽고 실행한다.
<p>Advantages</p>	<ul style="list-style-type: none"> • Scale-out for read, Geo-replication 가능 • Slow 네트워크나 네트워크 순간 단절에도 동작(Asynchronous 시) • LB for read, HA for read
<p>Dis-advantages</p>	<ul style="list-style-type: none"> • No Automatic Failover • No Guarantee on data consistency(Async) • No scalability on write • Master - Master로 구성도 가능하나 실제로는 DB 정합성에 문제가 있어 상용으로는 사용안함
<p>Applications</p>	<ul style="list-style-type: none"> • Master(write), slave(read) • Read transaction의 scale out을 통한 부하 분산용으로 가장 많이 쓰임



<기본개념도>



<응용 개념도>

01 DB 이중화, DR - MySQL replication (2/5)

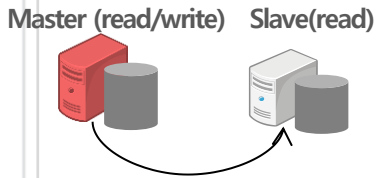
Replication 방식

- **Asynchronous** : slave의 acknowledge없이 transaction이 commit되는 것으로 간주(Default)
- **Semi-synchronous** : 최소한 하나의 slave로 부터 ACK를 확인해야 master의 transaction commit이 완료됨. MySQL 5.5 부터 제공
- **Synchronous** : MySQL Cluster 버전에서만 제공

Topologies

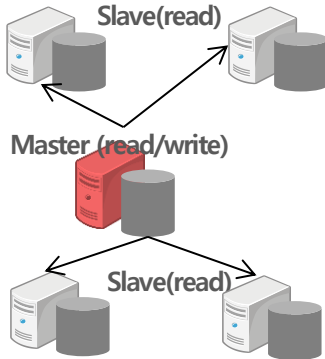
- 아래와 같이 다양한 topologies가 적용가능하며 한 개의 topology에 두 가지 이상의 방식을 혼용해서 사용하기도 한다.

Master-slave(1:1)



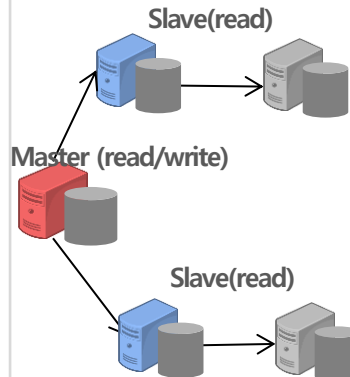
- 관리 용이, 기본 구조
- 최소 비용

Master-slave (1 : n)



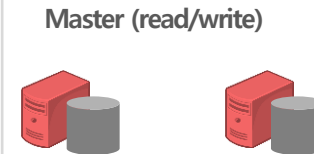
- Read load balancing
- Slave중 하나의 ack이 있어도 master는 동기화 된 것으로 client에게 보임
- 구성 비교적 용이
- read load balancing

Master-slaves of slaves



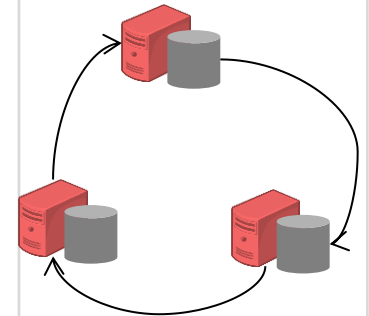
- Master만 read/write
- 1차 slave는 2차 slave의 master역할
- 설정 및 유지 보수 어려움
- 계층 구조로 동기화 어려움
- loosely coupled synch 구조
- 변경이나 수정이 많이 없는 페이지나 정보구성에 이용

Master-Master



- read & write load balancing
- write 성능 향상
- 데이터 정합성(Integrity) 깨지기 쉬움
- 상용환경에서는 잘 쓰이지 않음

Multi-master ring



- read & write load balancing
- write 성능 향상
- 데이터 정합성(Integrity) 깨지기 쉬움
- 구성하기 복잡함
- 복잡하고 관리가 어려움
- 이론적인 구조

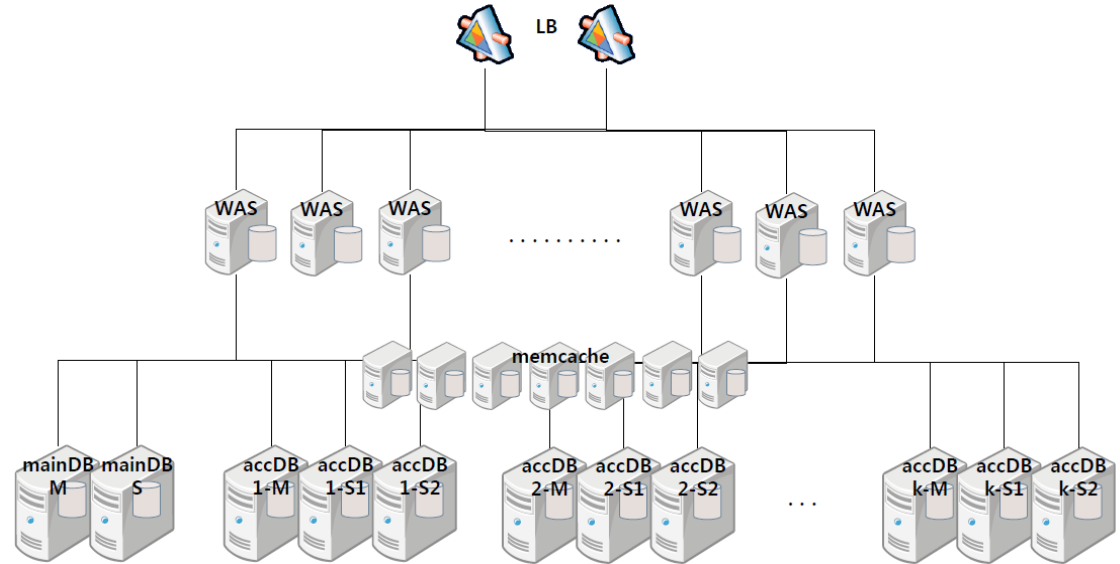
01 DB 이중화, DR - MySQL replication (3/5)

	Synchronous	Asynchronous	Semi-Synchronous
개념	<ul style="list-style-type: none"> master가 slave로 부터 acknowledge를 받았을 때 transaction이 commit 	<ul style="list-style-type: none"> slave의 acknowledge없이 transaction이 commit되는 것으로 간주(Default) 	<ul style="list-style-type: none"> slave에는 asynchronous, client에는 synchronous (delayed -acknowledgment commits)
장점	<ul style="list-style-type: none"> Master와 slave간에 데이터 동기화가 가장 잘 보장됨 	<ul style="list-style-type: none"> Transaction 완료시간이 빠름(slave의 commit여부와 상관없음) 여러 slave중 하나만 동기화 되어도 해당 slave는 read load 분산을 위해 client에 available하다 	<ul style="list-style-type: none"> 비교적 빠른 transaction 완료시간(세 방식 중 중간) DB client들에게 master -slave 간 데이터 동기화 보장 여러 slave중 하나만 동기화 되어도 해당 slave는 read load 분산을 위해 client에 available하다
단점	<ul style="list-style-type: none"> MySQL Cluster 버전에서만 제공 Transaction 완료시간이 세가지 방식 중 가장 느림 (성능) 	<ul style="list-style-type: none"> 평상시 데이터 동기화 지연 현상 발생 가능 장애 시 Master와 slave간의 데이터 불일치 가능성이 세가지 방식 중 가장 큼 장기간 운영시 master - slave간 데이터 불일치 발생 	<ul style="list-style-type: none"> Transaction 완료시간은 asynchronous방식보다 느림 Master 장애시 데이터 동기화는 synchronous 방식보다는 잘 보장되지 않음

01 DB 이중화, DR - MySQL replication (4/5)

Replication in ucloud

ucloud 고객인 게임전문회사 A사가 replication과 sharding을 이용하여 시스템을 확장한 사례



응용 기술

scale-out(WAS)

LB 이중화
WAS의 scaling out을 통한 LB

memcache

DB에 대한 read cache역할로
DB 쿼리에 대한 성능 향상

replication

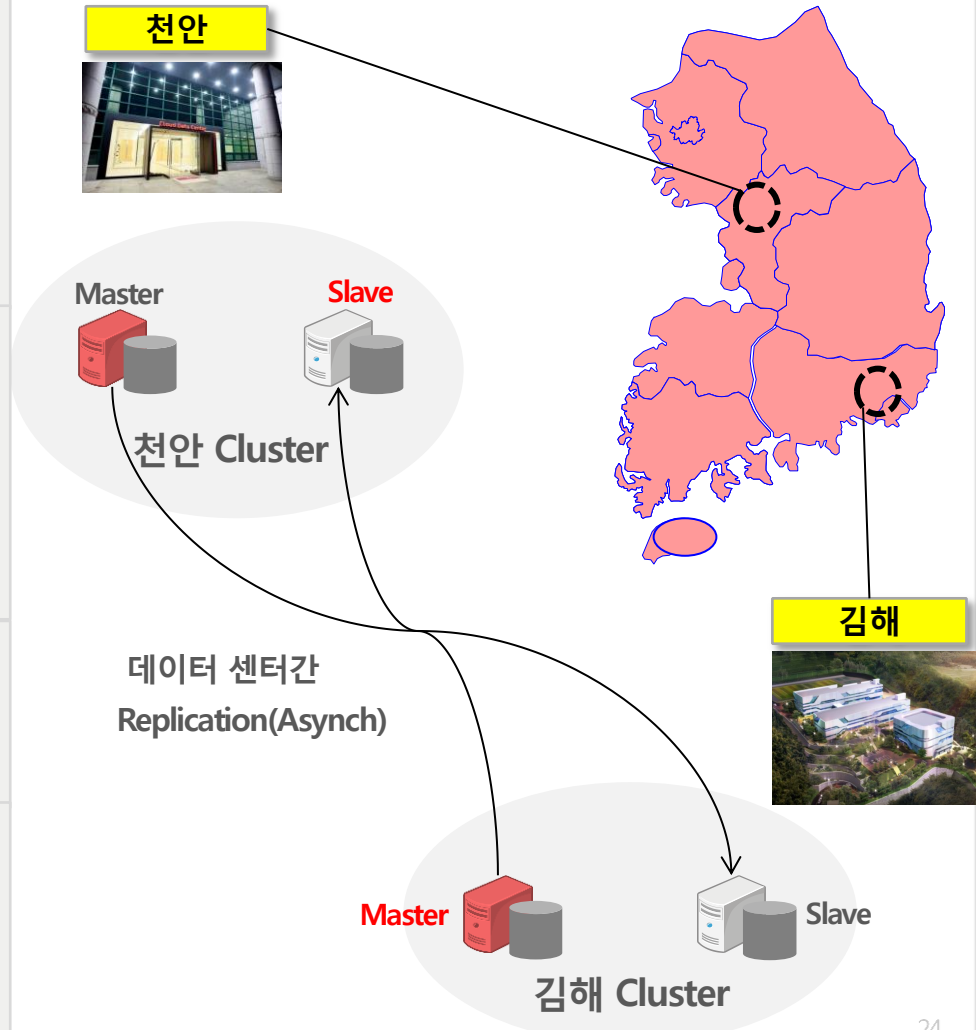
모든 DB서버를 1 Master, 2
Slave로 구성하여 read
transaction을 부하분산

sharding

Account DB의 테이블을 수평
적으로 분할하여 부하를 분
산

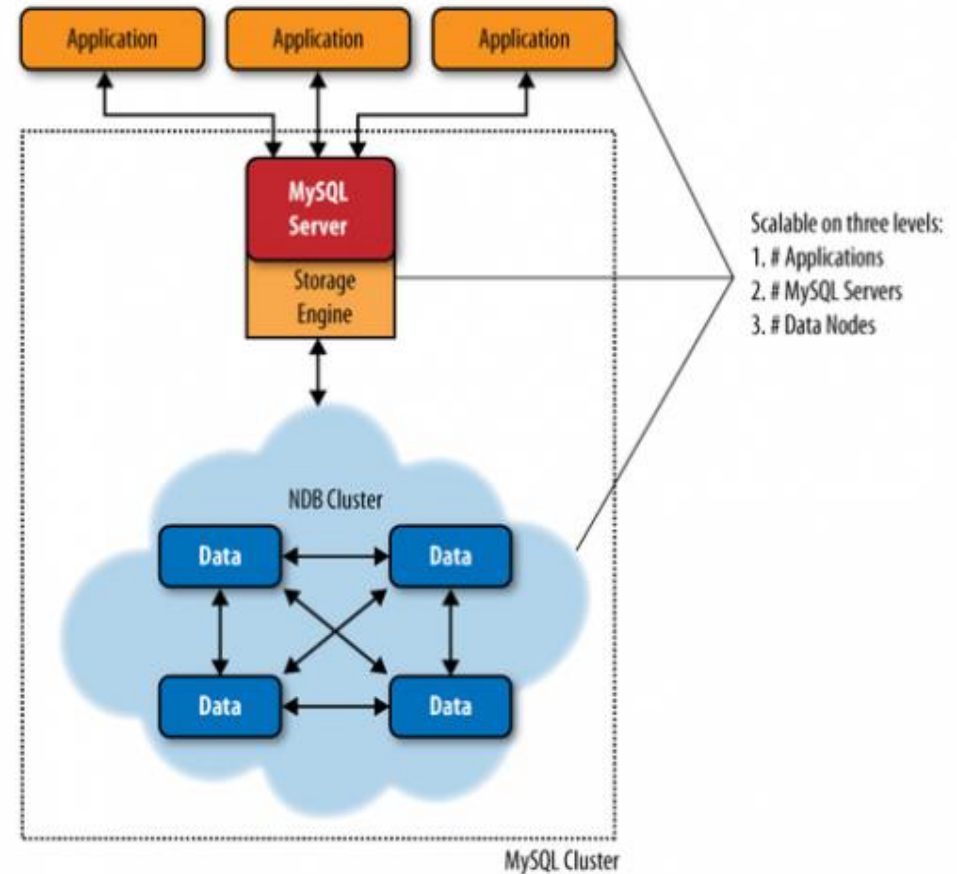
01 DB 이중화, DR - MySQL geo-replication (5/5)

<p>구성방법</p>	<p>MySQL Cluster-replication 구성</p> <ul style="list-style-type: none"> • 데이터 센터내에 MySQL cluster 구성 • 데이터 센터간 replication 구성 <p>Replication 구성</p> <ul style="list-style-type: none"> • 데이터 센터간 replication으로만 구성 • 기타 여러가지 구성 조합이 가능
<p>이중화 원리</p>	<ul style="list-style-type: none"> • 같은 데이터 센터 내에서는 synchronous 구성을 통한 Cluster를 구성한다 • 데이터 센터간에는 asynchronous replication을 통한 데이터 동기화를 통해 DR 구성이 가능하다. • Cluster간 동기화는 MySQL에서 제공하는 collision detection 및 resolution 기능 이용
<p>Advantages</p>	<ul style="list-style-type: none"> • 사용자에게 보다 가깝게 데이터를 위치시켜 latency 영향을 최소화 • Disaster recovery 구성으로 안정성 강화
<p>Dis-advantages</p>	<ul style="list-style-type: none"> • 데이터센터간 network latency가 bottleneck • Master 노드 장애시 asynchronous replication에 따른 데이터 동기화(consistency) 문제 발생 • Cluster NDB의 성능 문제, 고가의 라이선스 비용



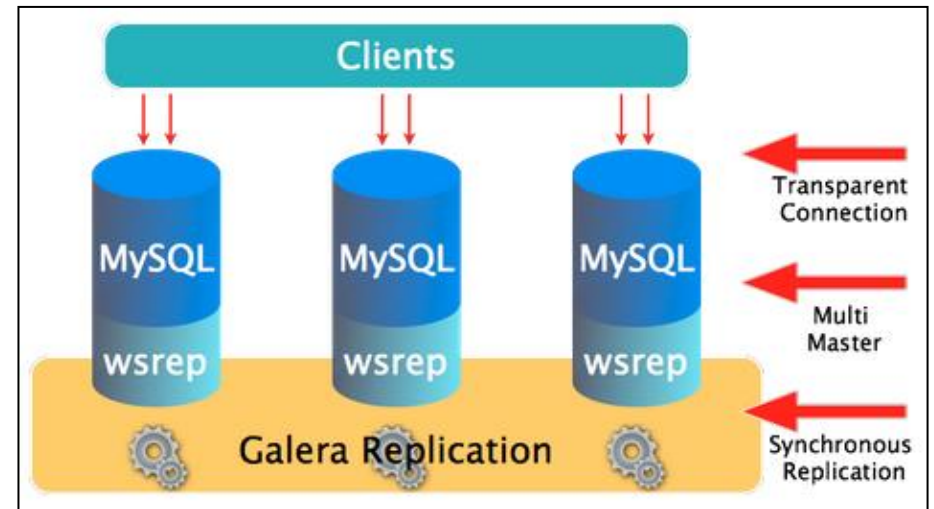
02 DB 이중화, DR - MySQL Clustering

<p>구성요소</p>	<p>Applications</p> <ul style="list-style-type: none"> MySQL과 통신 <p>MySQL servers</p> <ul style="list-style-type: none"> SQL commands 처리 및 NDB와 통신 <p>NDB cluster</p> <ul style="list-style-type: none"> Query 처리와 결과를 MySQL에 return
<p>이중화 원리</p>	<ul style="list-style-type: none"> Application, MySQL, NDB가 독립적으로 scale out 가능한 구조 Data가 peer Data Nodes Cluster 안에서 Synchronous하게 복제된다. 각각의 데이터 노드는 다른 데이터 노드들과 모두 연결되어 있고 데이터는 여러 개의 데이터 노드에 저장된다.
<p>Advantages</p>	<ul style="list-style-type: none"> No Single Point of Failure 운영 중 테이블 변경, 데이터 노드 추가 가능 HA, High Performance(Active-Active) Automatic failover
<p>Dis-advantages</p>	<ul style="list-style-type: none"> NDB의 성능이 느려서 상용으로는 잘 쓰이지 않음(데이터 분산 저장으로 인한 Join query 등이 느림) 고가의 라이선스 비용으로 상용으로 사용하는 예는 거의 없음



03 DB 이중화, DR – Galera Cluster

<p>구성요소</p>	<p>DB</p> <ul style="list-style-type: none"> MySQL or Maria DB 사용 <p>wsrep</p> <ul style="list-style-type: none"> wsrep API로 노드 간 통신 <p>Galera Replication</p> <ul style="list-style-type: none"> Synchronous 방식의 Replication을 제공
<p>이중화 원리</p>	<ul style="list-style-type: none"> 동기 방식의 데이터 복제 Active-Active 방식의 다중 마스터 구성 클러스터 내 노드 자동 컨트롤 특정 노드 장애 시 자동으로 해당 노드 제거 자동으로 신규 노드 추가 가능 행 단위로 완벽한 병렬적 리플리케이션 기존의 MySQL 클라이언트 방식으로 동작
<p>Advantages</p>	<ul style="list-style-type: none"> 마스터/슬레이브 간 데이터 동기화 지연 없음 트랜잭션 유실이 없음 읽기/쓰기 모두 확장이 가능 클라이언트에 대기 시간이 길지 않음
<p>Dis-advantages</p>	<ul style="list-style-type: none"> 노드 추가 시마다 데이터 복제로 부하 발생 서버 간 그룹 커뮤니케이션 부하 발생 모든 노드에 동일한 데이터를 유지로 공간 낭비



04 DB 이중화, DR - Sharding

SID (PK)	cust	telno	addr
10333304561	이만수	010-1111-2222	양천구 목동
10293837645	선동열	010-2222-3333	강남구 역삼동
19746478585	이승엽	010-5555-2222	성동구 옥수동
18464525272	최동원	010-4455-7777	광진구 구의동



SID (PK)	cust	telno	addr
10333304561	이만수	010-1111-2222	양천구 목동
19746478585	이승엽	010-5555-2222	성동구 옥수동



SID (PK)	cust	telno	addr
10293837645	선동열	010-2222-3333	강남구 역삼동
18464525272	최동원	010-4455-7777	광진구 구의동



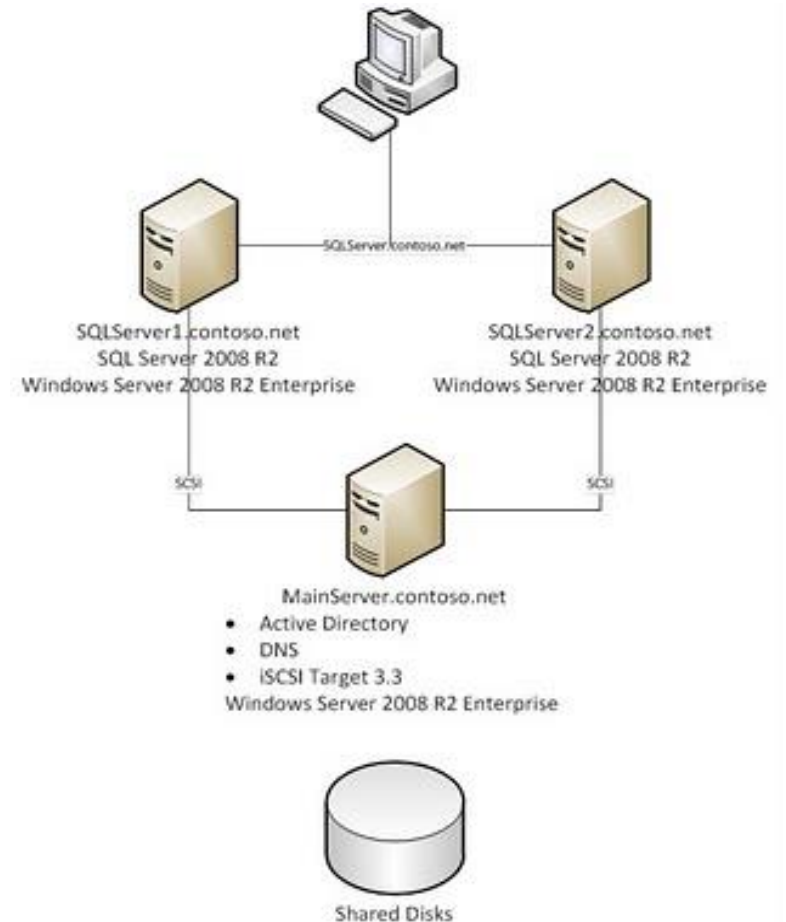
Sharding	<ul style="list-style-type: none"> 같은 테이블을 여러 DB인스턴스로 수평 분할하는 방식으로 어플리케이션에서 구현하거나 sharding 플랫폼을 통해 구현한다.(MySQL proxy기반의 Spock Proxy와 Gizzard, Cubrid sharding 등이 있다.)
Advantage	<ul style="list-style-type: none"> search, read/write load 분산 Scale-out에 매우 유리한 cloud-friendly architecture
Disadvantage	<ul style="list-style-type: none"> 두 개이상의 shard에 대한 직접적인 join연산 불가 shard key(column)값은 업데이트 안됨 두 개 이상의 shard에 대한 transaction 불가

05 DB 이중화, DR - MySQL with Cluster, Replication

	 Cluster	 Replication
개념	<ul style="list-style-type: none"> • Transaction-safe • Real time 복제 • Active-Active 구조 • App, MySQL, NDB가 독립적인 Scale-out 	<ul style="list-style-type: none"> • Master – Slave 구조 • Active-standby • Active-Active(write-read)
장점	<ul style="list-style-type: none"> • Clustering으로 DB서버 HA 보장 • Scale-out • True DB redundancy 	<ul style="list-style-type: none"> • 데이터 이중화 • Scale-out for read
단점	<ul style="list-style-type: none"> • 고가의 라이선스 비용 • 네트워크 기반의 동기화 및 복제에 따른 성능저하로 상용환경에서는 많이 쓰이지 않음 	<ul style="list-style-type: none"> • No automatic failover • Data consistency Not guaranteed (Asynchronous)
용도	<ul style="list-style-type: none"> • LB, HA,가 필요한 application 	<ul style="list-style-type: none"> • Read-intensive한 application

06 DB 이중화, DR - MS-SQL with Clustering (MSCS)

개념	<ul style="list-style-type: none"> • SQL 서버 단위의 clustering • 공유 스토리지 보유 • Active-Standby 구성 모두 가능
이중화 원리	<ul style="list-style-type: none"> • Cluster로 구성된 서버들간에 heartbeat을 체크하여 Active-Standby 구조라면 Standby 서버로 automatic failover 된다.
Advantages	<ul style="list-style-type: none"> • Clustering으로 DB서버 HA 보장
Dis-advantages	<ul style="list-style-type: none"> • 공유 스토리지는 scale-out 할 수 없음 • DB 서버만 HA, LB 됨 • AD 서버가 있어야 함
보완	<ul style="list-style-type: none"> • 공유 스토리지는 별도로 이중화 등을 구성할 수 있다. • ucloud 환경에서 공유 스토리지는 ucloud NAS로 구성 가능함



07 DB 이중화, DR - MS-SQL mirroring

개념

- Principle server만 서비스, Mirror server는 principle server 장애시 failover(automatic/manual)
- DB의 HA를 확장하는 개념(스토리지 포함)

이중화 원리

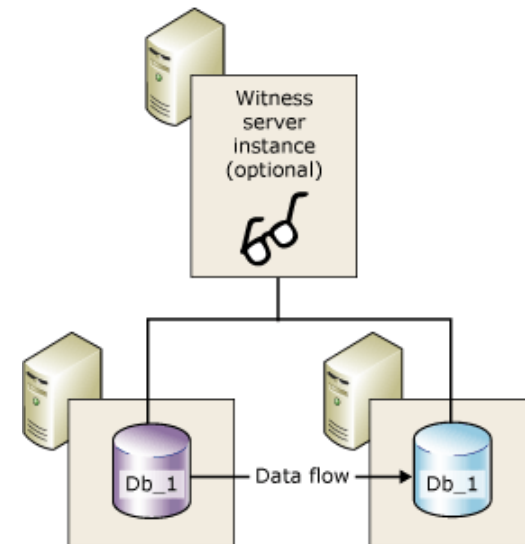
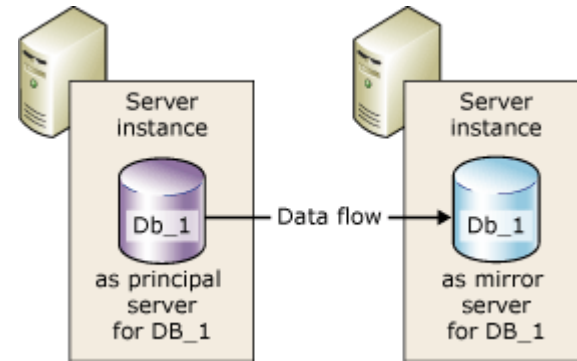
- Principle server가 Log record를 mirror서버로 보내서 복제를 하며 synch 방식과 asynch방식이 있다.
- Synchronous for consistency
- Asynchronous for performance

Advantages

- Synchronization replication의 경우 automatic failover 가능 (auto-failover위해 Witness 서버 필요)
- DB와 스토리지 HA (공유스토리지가 아니라 복제 스토리지 사용)

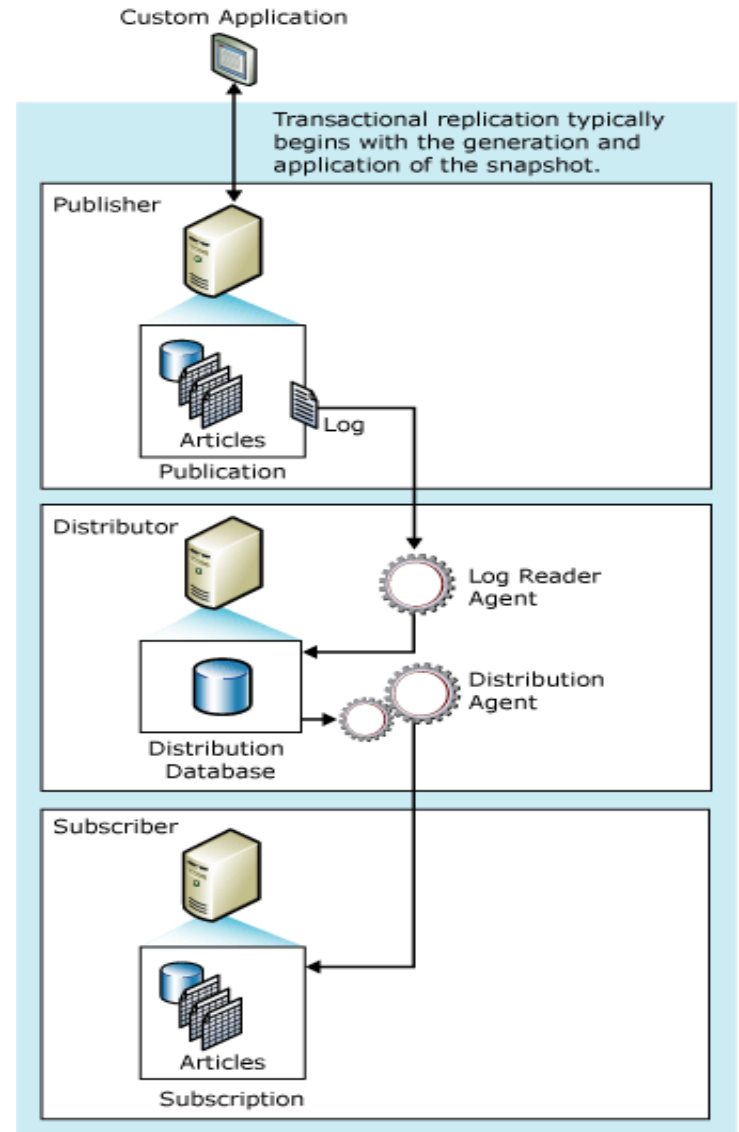
Dis-advantages

- No LB for DB 서버
- 평상시 Principle만 가용하고 장애시는 mirror만 가용함



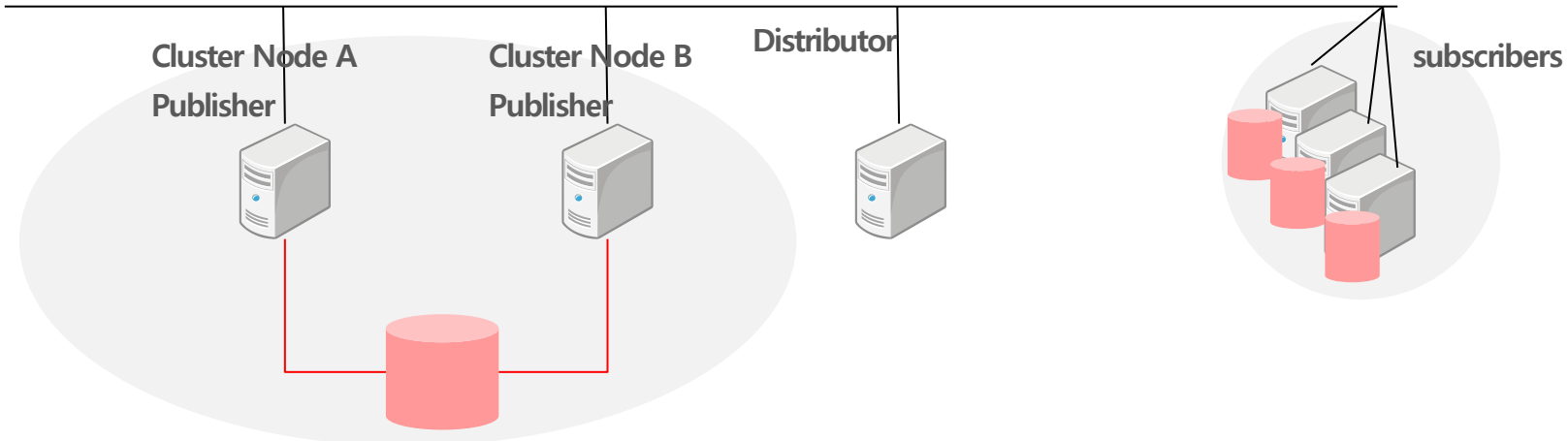
08 DB 이중화, DR - MS-SQL replication

<p>개념</p>	<ul style="list-style-type: none"> • Publisher가 transaction에 대해 생산한 로그를Distributor가 해석하고 이를 바탕으로 subscriber(s)들에게 제공하여 Publisher와 subscriber(s)사이에 consistency 유지
<p>이중화 원리</p>	<ul style="list-style-type: none"> • Transactional: 최초 snapshot 동기화 후 transaction에 의한 동기화 • Merge: 최초 snapshot 동기화 그 후 Distributor가 Publisher와 Subscriber(s)의 Update을 동기화 • Snapshot: 주기적으로 Publisher에서 Subscriber(s)로 snapshot 보내서 동기화
<p>Advantages</p>	<ul style="list-style-type: none"> • Subscriber(s)를 read-only로 구성하여 LB 가능 • Subscriber(s)가 write, read LB가능(Only Merge replication)
<p>Dis-advantages</p>	<ul style="list-style-type: none"> • No Automatic Failover • No Guarantee on data consistency(Async) • No scalability on write






09 DB 이중화, DR - MS-SQL Clustering with replication

<p>개념</p>	<ul style="list-style-type: none"> MS-SQL Cluster(DB) + replication(Storage) 	<p>Advantages</p>	<ul style="list-style-type: none"> 가용성: Cluster failover 확장성: read 데이터를 subscriber로 분산시킴으로써 성능향상을 꽤 할 수 있다.
<p>이중화 원리</p>	<ul style="list-style-type: none"> Publisher 역할을 할 서버를 Cluster로 구성하여 가용성을 높인다. Subscriber를 추가하여 스토리지에 대한 확장성을 높인다. Publisher는 write에 대한 작업을 담당하고 subscriber는 read 작업을 담당하도록 어플리케이션에서 설계해야 한다 	<p>Dis advantages</p>	<ul style="list-style-type: none"> No write storage HA(공유 스토리지 사용) No write storage redundancy Cluster의 공유 스토리지 장애시 Master역할을 subscriber로 수작업 전환 필요

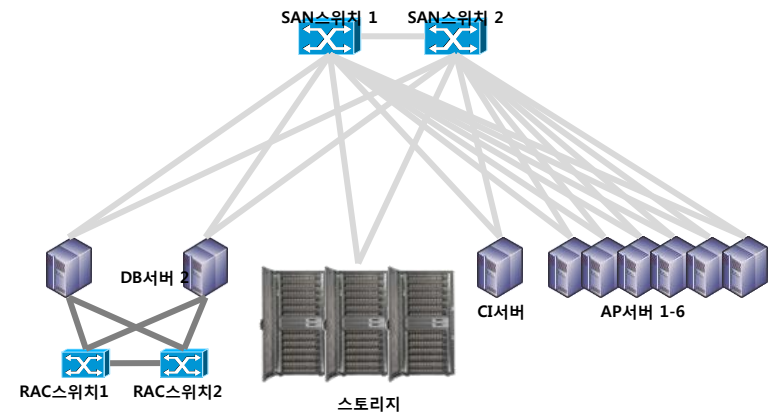
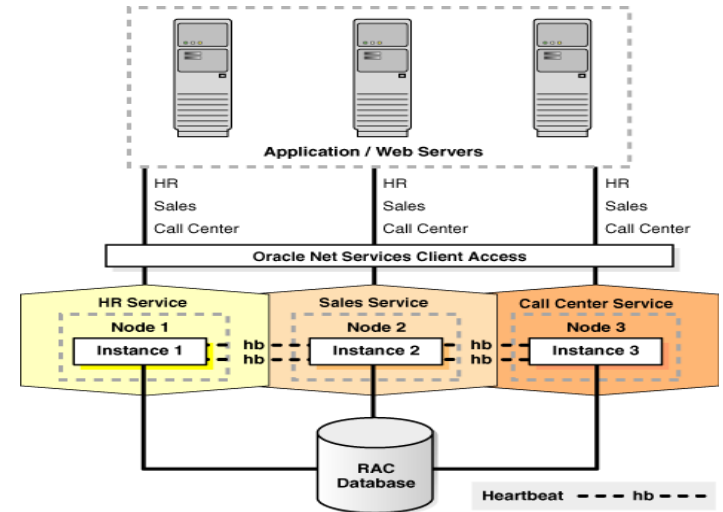


DB 이중화, DR - MS-SQL Clustering, Replication & Mirroring

	 Clustering	 Mirroring	 Replication
개념	<ul style="list-style-type: none"> DB server HA 제공 공유 스토리지 보유 Active-Passive 구성 	<ul style="list-style-type: none"> 하나의 데이터베이스에 대한 2개의 복사본을 서로 다른 SQL server에 보관하는 것 Principle server만 서비스, Mirror server는 principle server 장애시 failover(automatic/manual) Principle-Mirror Synchronous/Asynchronous 	<ul style="list-style-type: none"> 같은 데이터베이스에 대한 복사본을 여러 개의 다른 서버에 보관하는 것 Publisher-Subscriber(s) Active-Standby 구조
장점	<ul style="list-style-type: none"> DB 서버 레벨의 Clustering Clustering으로 DB서버 HA 보장 	<ul style="list-style-type: none"> HA for DB server (Automatic Failover) 	<ul style="list-style-type: none"> LB for read subscriber(s) LB for subscriber(s) read & write(Merge replication) Replication을 DB, table 등의 단위로 선택가능
단점	<ul style="list-style-type: none"> 공유 스토리지는 별도로 이중화 구축 DB 서버만 HA, LB 됨 공유 스토리지는 scale-out 할 수 없음 	<ul style="list-style-type: none"> Mirror DB 정상 동작 시 접근 불가 No LB 	<ul style="list-style-type: none"> No automatic failover
용도	<ul style="list-style-type: none"> DB Server의 가용성이 중요하나 스토리지 성능의 scale-out이 필요없는 곳 	<ul style="list-style-type: none"> DB 가용성 향상이 필요하나 DB 및 스토리지의 scale-out은 필요없는 곳 	<ul style="list-style-type: none"> Read 성능 scale-out

11 DB 이중화, DR - Oracle RAC(Real Application Clusters)

<p>개념</p>	<ul style="list-style-type: none"> Cluster구성의 shared-nothing, shared disk의 한계를 shared cache 아키텍처로 극복하여 DB의 scalability와 HA를 동시에 보장한다
<p>구성요소</p>	<ul style="list-style-type: none"> Oracle instances: 각 호스트에 DB 인스턴스 Oracle clusterware: cluster 구성이 필요한 기능들을 제공하고 관리하며 oracle DB와 독립적으로 설치될 수도 있다.
<p>이중화 원리</p>	<ul style="list-style-type: none"> 호스트별로 DB인스턴스를 생성하고 Clusterware를 이용하여 각 호스트들을 관리하고 메모리를 공유하는 구조를 가짐으로써 DB의 HA와 LB를 동시에 가능하게 한다.
<p>Advantages</p>	<ul style="list-style-type: none"> LB for DB server (throughput 향상) Clustering으로 DB서버 HA 보장
<p>Dis-advantages</p>	<ul style="list-style-type: none"> 공유 스토리지에 대해 scale-out 불가 공유 스토리지가 Single Point of Failure : 별도로 이중화 필요 성능을 고려한 확장가능한 Cluster 수는 최대 6개의 노드 정도이다.(3개 이상부터 30%정도씩 성능이 떨어짐)

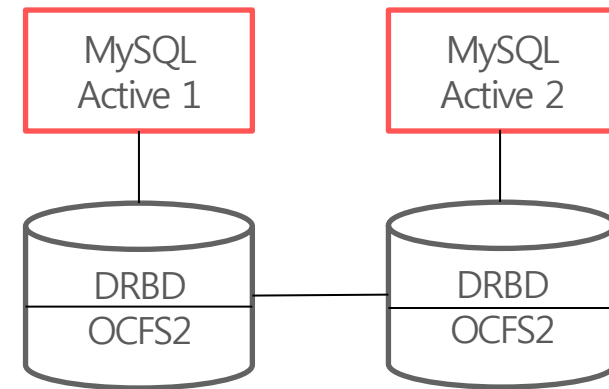


Shared nothing : 분산 컴퓨팅 아키텍처에서 여기서는 이중화 DB아키텍처에서 노드간에 메모리를 공유하지 않고 독자적인 메모리를 가지는 것

Shared disc : 디스크 즉 스토리지를 공유하는 것으로 스토리지가 기존 MS-SQL clustering은 스토리지가 single point of failure가 된다.

12 DB 이중화, DR – Dual Primary Cluster File System

Use Case	<ul style="list-style-type: none"> Oracle RAC 용 스토리지 이중화 구성 MySQL 의 스토리지 이중화 구성
구성요소	<p>DRBR</p> <ul style="list-style-type: none"> 블록 레벨 데이터 복제 <p>OCFS2</p> <ul style="list-style-type: none"> 읽기/쓰기 동작의 동시성 보장 동시성 제어시 Locking 관리
이중화 원리	<ul style="list-style-type: none"> 동기 방식의 데이터 복제 Active-Active 방식의 다중 마스터 구성 특정 노드 장애 시 STONISH 스크립트 사용필요
Pros	<ul style="list-style-type: none"> 개별 인스턴스는 로컬 디스크 사용효과 마스터/슬레이브 간 데이터 동기화 지연 없음 트랜잭션 유실이 없음 클라이언트에 대기 시간이 길지 않음
Cons	<ul style="list-style-type: none"> 노드 추가 시마다 데이터 복제로 부하 발생 서버 간 그룹 커뮤니케이션 부하 발생 모든 노드에 동일한 데이터를 유지로 공간 낭비



상세 구성 방법

<http://developer.ucloudbiz.olleh.com/blog/serverUse/DRBD--OCFS2---dual-primary-cluster-filesystem-/>

13 DB 이중화, DR - Cluster 기술 비교

	 MySQL	 MS-SQL	 Oracle RAC
개념	<ul style="list-style-type: none"> • Transaction-safe • Real time 복제 • Active-Active 구조 • App, MySQL, NDB가 독립적인 Scale-out • Shared nothing 구조(메모리 공유 안함) 	<ul style="list-style-type: none"> • 하나의 데이터베이스에 대한 2개의 복사본을 서로 다른 SQL server에 보관하는 것 • Shared Nothing 구조 	<ul style="list-style-type: none"> • Cluster구성의 shared-nothing, shared disk의 한계를 shared cache 아키텍처로 극복하여 DB의 scalability와 HA를 동시에 보장한다 • Shared Everything 구조로 메모리 공유
장점	<ul style="list-style-type: none"> • No Single Point of Failure • Clustering으로 DB서버 HA 보장 • 운영 중 테이블 변경, 데이터 노드 추가 가능 • LB for DB server cluster(throughput 향상) 	<ul style="list-style-type: none"> • Clustering으로 DB서버 HA 보장 	<ul style="list-style-type: none"> • Clustering으로 DB서버 HA 보장 • LB for DB server cluster(throughput 향상)
단점	<ul style="list-style-type: none"> • 네트워크 기반의 NDB 동기화 및 복제에 따른 성능저하로 상용환경에서는 많이 쓰이지 않음 	<ul style="list-style-type: none"> • 공유 스토리지에 대해 scale-out 불가 • 공유 스토리지가 Single Point of Failure : 별도로 이중화 필요 	<ul style="list-style-type: none"> • 공유 스토리지에 대해 scale-out 불가 • 공유 스토리지가 Single Point of Failure : 별도로 이중화 필요 • 고가의 라이선스 비용
용도	<ul style="list-style-type: none"> • 소용량의 데이터로 간단한 쿼리를 실행하는 App. 웹 사이트 세션 저장, 파일 스토리지 메타 데이터 저장 등 	<ul style="list-style-type: none"> • 기업용 DB로 상용으로 많은 reference가 있다. 스토리지는 SAN이나 스토리지 컨트롤러 이중화, RAID 구성 등으로 보완할 수 있다. 	<ul style="list-style-type: none"> • 다소 고가이기는 하지만 안정적인 DB환경을 원하는 기업고객들이 많이 사용하고 있다.

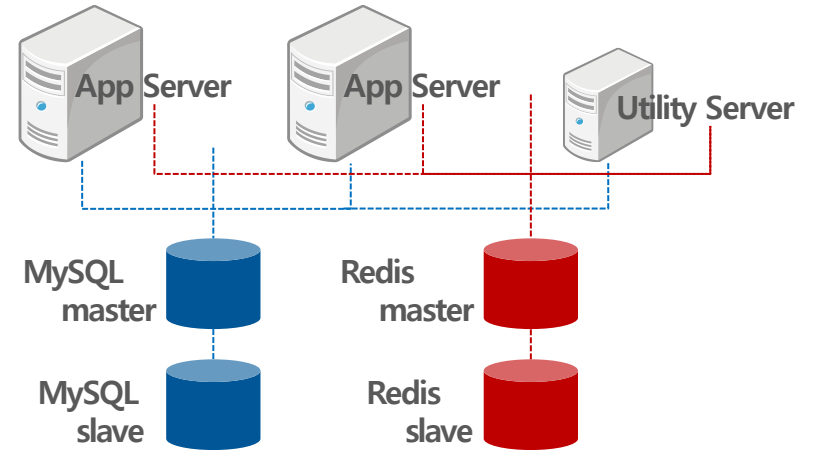
14 DB 이중화, DR - Redis

<http://developer.ucloudbiz.olleh.com/blog/cloudstack/ucloud-Redis--/>

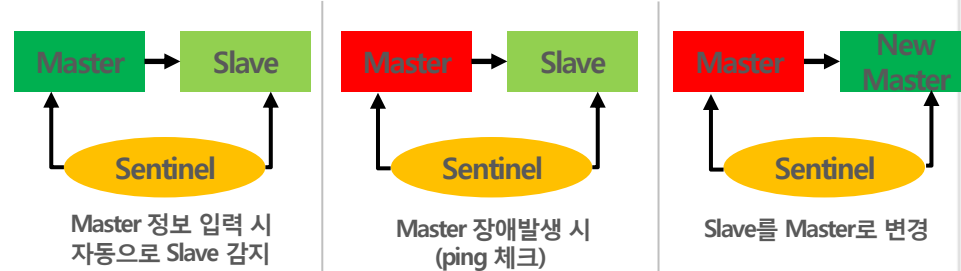
<h3>주요특징</h3>	<ul style="list-style-type: none"> In-memory key-value 형식의 데이터 저장을 지원하여 매우 빠른 write/read 속도를 보장 다양한 data type을 지원한다.
<h3>이중화 원리</h3>	<ul style="list-style-type: none"> Master-Slave로 구성하며, 'sentinel' 모드를 작동시켜 Redis instance의 상태 모니터링 및 auto failover를 가능하게 한다. Key 값을 hashing하여 key/value pair가 저장될 shard를 선택하는 방식을 이용하여 수평적으로 확장이 가능하다.
<h3>Advantages</h3>	<ul style="list-style-type: none"> In-memory: 빠른 IOPS 처리 Persistence: 데이터를 disk에 저장하여 데이터 유실방지, 저장방식으로는 snapshot과 AOF 방식 지원 Sharding/replication이 용이: Query Off Loading 기법을 사용하여 성능향상 가져옴
<h3>Dis-advantages</h3>	<ul style="list-style-type: none"> 메모리보다 큰 데이터는 부적합함 Clustering 기능 지원 안함 (2013.4Q 지원예정) Sharding을 일일이 지정해줘야 함

Sentinel: Redis의 분산 시스템으로 redis-server에 -sentinel 옵션을 사용하여 sentinel 프로세스를 실행
Query Off Loading: master node는 write only, slave node는 read only로 사용하는 방법
AOF(Append on File): Redis의 모든 write/update 연산을 log파일에 기록하여 서버가 down 되더라도, 데이터 유실이 발생하지 않음

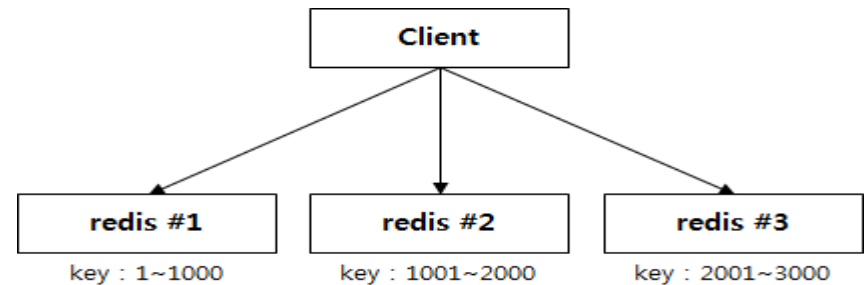
Redis 구성도



Sentinel 동작원리

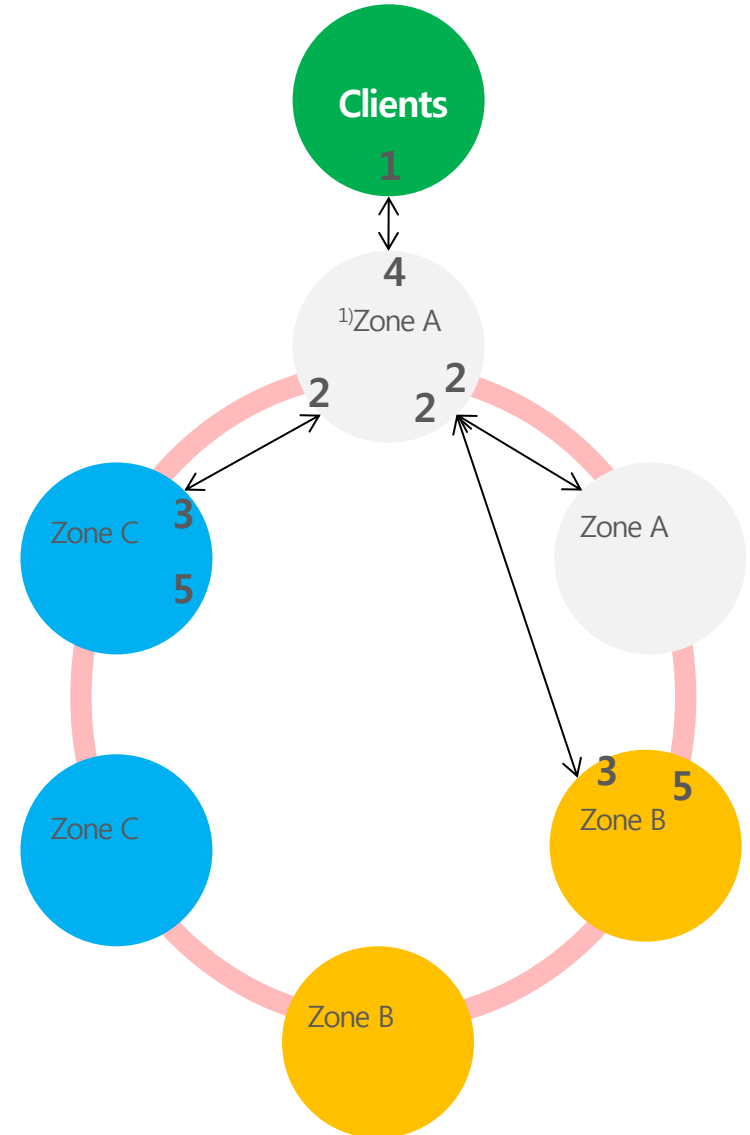


Sharding을 통한 용량확장



15 DB 이중화, DR - Cassandra

<p>Cassandra</p>	<ul style="list-style-type: none"> • Big data의 분산 처리를 위한 데이터 베이스로 유연한 스키마 구성(row마다 속성을 다르게 구성가능)이 하고 key-value 구조로 빠른 성능을 보임
<p>이중화 원리</p>	<ul style="list-style-type: none"> • Redundancy 구성은 다음과 같이 이뤄진다 <ol style="list-style-type: none"> 1. clients가 write operation을 cassandra의 어느 노드에 한다. 2. Coordinator 노드가 데이터를 노드들과 다른 존에 복제한다. 3. 노드들이 ack을 Coordinator들에 return한다. 4. Coordinator가 ack을 노드들에 return한다. 5. 데이터가 internal commit log 디스크에 기록된다.
<p>Advantages</p>	<ul style="list-style-type: none"> • Fault-tolerant with replication • No single point of failure • Read/write scalability (여러 노드가 장애 시에도 남아 있는 노드에서 서비스 가능)
<p>Dis-advantages</p>	<ul style="list-style-type: none"> • Join operation 지원 안됨 • 새 노드 추가시 데이터 재분배 시간 소요 • 메모리보다 큰 object들은 저장에 어려움

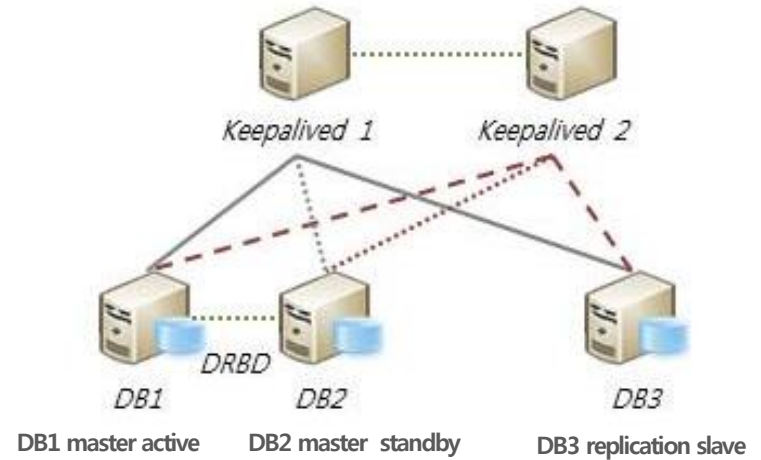


1) 그림에서 zone은 data center 단위라고 바꿔서 생각해도 됨

16 DB 이중화, DR - Replication, DRBD and keepalived

응용사례

ucloud 고객인 D그룹 replication, DRBD와 keepalived를 이용한 시스템 구축 사례



MySQL replication	<ul style="list-style-type: none"> • DB1을 Master DB3를 slave로 replication 구성 • No auto-failover • DB3로 수평적 확장 및 read load 분산
DRBD	<ul style="list-style-type: none"> • Block device 단위 복제를 위해 DRBD를 사용하여 DB1의 데이터를 DB2와 동기화 하였다.
Keepalived	<ul style="list-style-type: none"> • keepalived간에는 heartbeat를 서로 송수신하여 문제 발생시 failover 기능 지원 • Keepalived1은 DB1과 DB2간의 active-standby auto-failover • Keepalived2은 DB1과 DB3간의 LB를 통한 read load 분산

00 Reference

책

High Performance MySQL by Baron Schwartz외 다수
24시간 365일 서버/인프라를 지탱하는 기술 (이토 나오야 등, Jpub)

웹 사이트

<http://developer.ucloudbiz.olleh.com/blog/cloudstack/MSSQL-Cluster--Replication-Read---/?query=cluster>
<http://blogs.technet.com/b/meamcs/archive/2010/12/04/sql-mirroring-amp-replication.aspx>
<http://www.slideshare.net/thavamuni/high-availability-with-mysql-presentation-634755>
<http://social.msdn.microsoft.com/Forums/en-US/sqlreplication/thread/c7fb80ed-c881-4111-a4eb-2f1fbd6534ce>
<http://www.mysqlperformanceblog.com/2008/04/28/mysql-replication-vs-drbd-battles/>
https://blogs.oracle.com/MySQL/entry/where_would_i_use_mysql
<http://lists.mysql.com/cluster/2154>
<http://www.mysqlperformanceblog.com/2008/04/28/mysql-replication-vs-drbd-battles/>
<http://rsync.samba.org/>
<http://dev.kthcorp.com/2011/07/28/redis-buildingfast-lightweight-webapp/>
<http://redis.io/>
<http://us.sios.com/windows-replication-availability-software-smb>
<http://en.wikipedia.org/wiki/Rsync>
<http://answers.oreilly.com/topic/1705-the-value-of-asynchronous-mysql-replication/>
http://www.facebook.com/note.php?note_id=23844338919&id=9445547199&index=0
https://blogs.oracle.com/MySQL/entry/synchronously_replicating_databases_across_data
<http://dbperf.wordpress.com/tag/mysql-cluster-geographical-replication/>
<http://www.slideshare.net/datacharmer/advanced-mysql-replication-techniques>
<http://www.datastax.com/docs/0.8/introduction/index>
<http://dev.kthcorp.com/2011/06/29/what-is-cassandra-dbms/>

Thank you

