



ORACLE[®]

MySQL Architecture Design Patterns for Performance, Scalability, and Availability

Brian Mizejewski
Principal Manager Consulting

Alexander Rubin
Principal Consultant

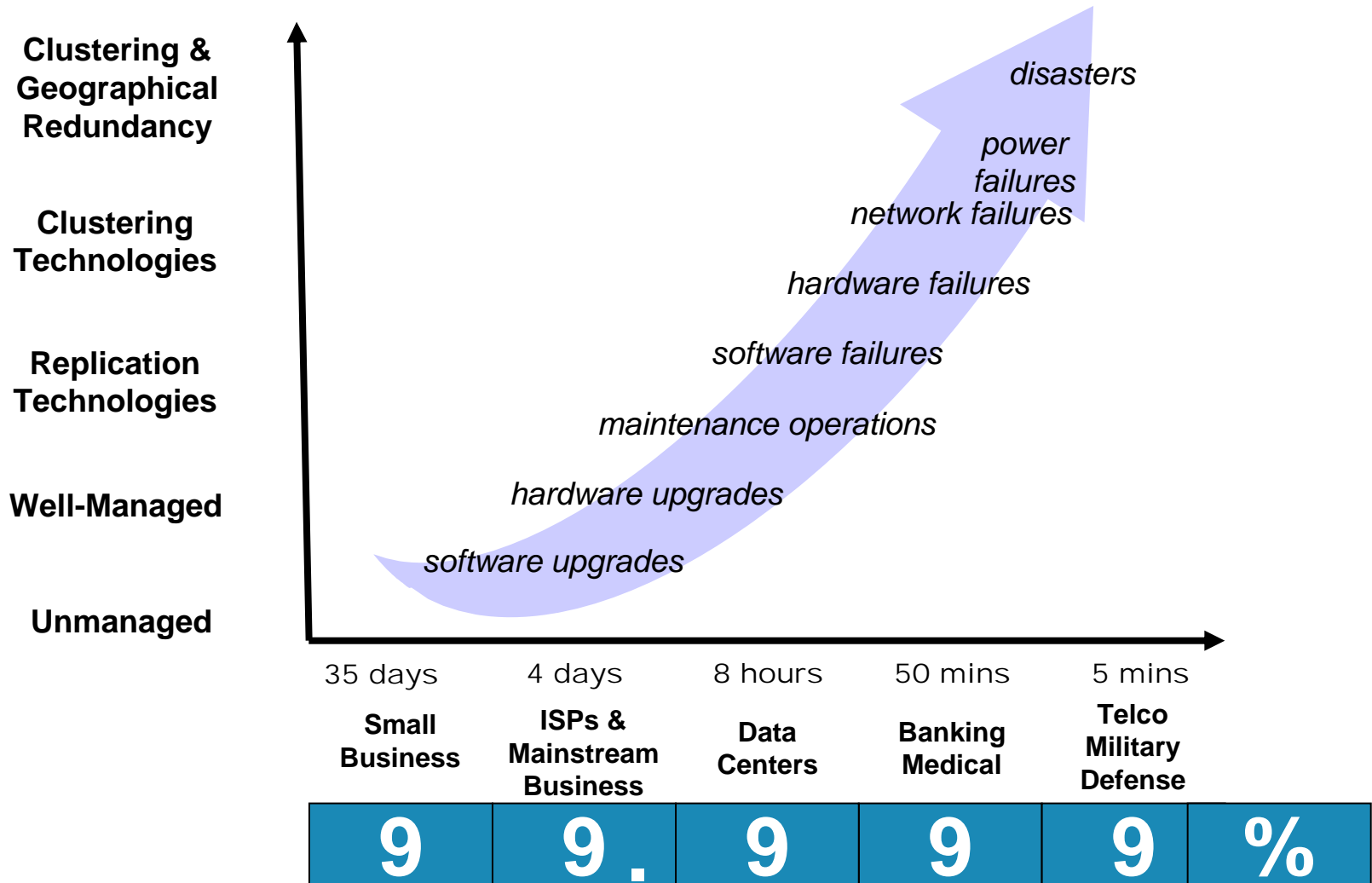
Agenda

- HA and Scale-Out
- Architectural Tools
 - MySQL Replication
 - DRDB Clustering
 - MySQL Cluster
- Design Patterns
 - Simple Master-Slave
 - Master and Many Slaves
 - Master-Master
 - DRBD Pair
 - DRBD Master with Many Slaves
 - MySQL Cluster
- Scale-out/Sharding

MySQL High Availability

- What is High Availability?
 - Availability of resources in a computer system
- Continuous Availability
 - Non-stop service
 - No disruption of service even during a fail over

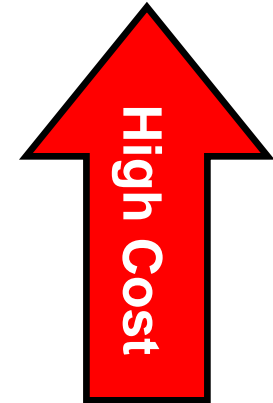
The Five 9s of Availability



Scale-Up vs. Scale-Out

- ***Scale-Up***

- Vertical
- Expensive SMP hardware
- Proprietary software
- Platform lock-in
- “Fork Lift” to increase capacity & performance

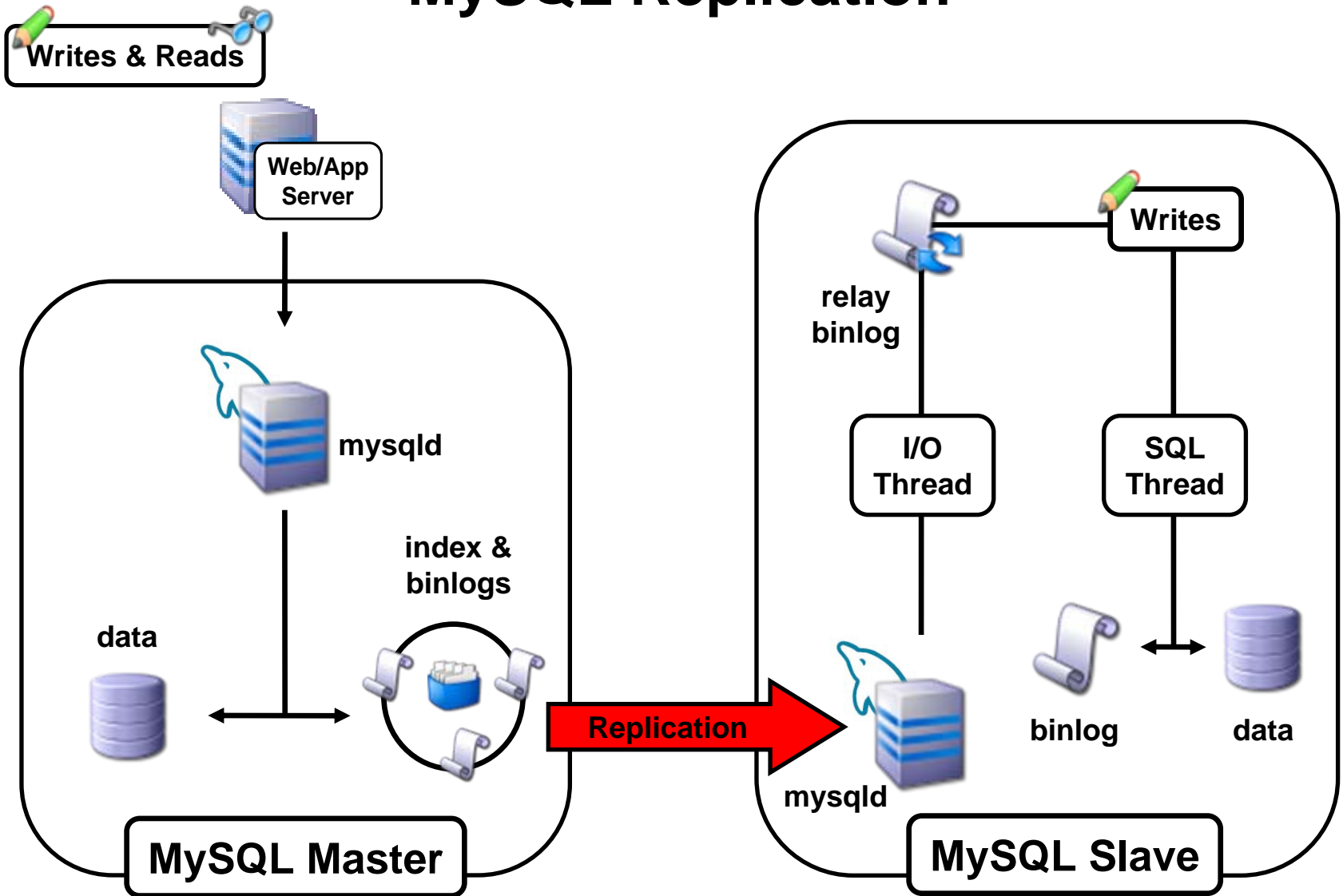


- ***Scale-Out***

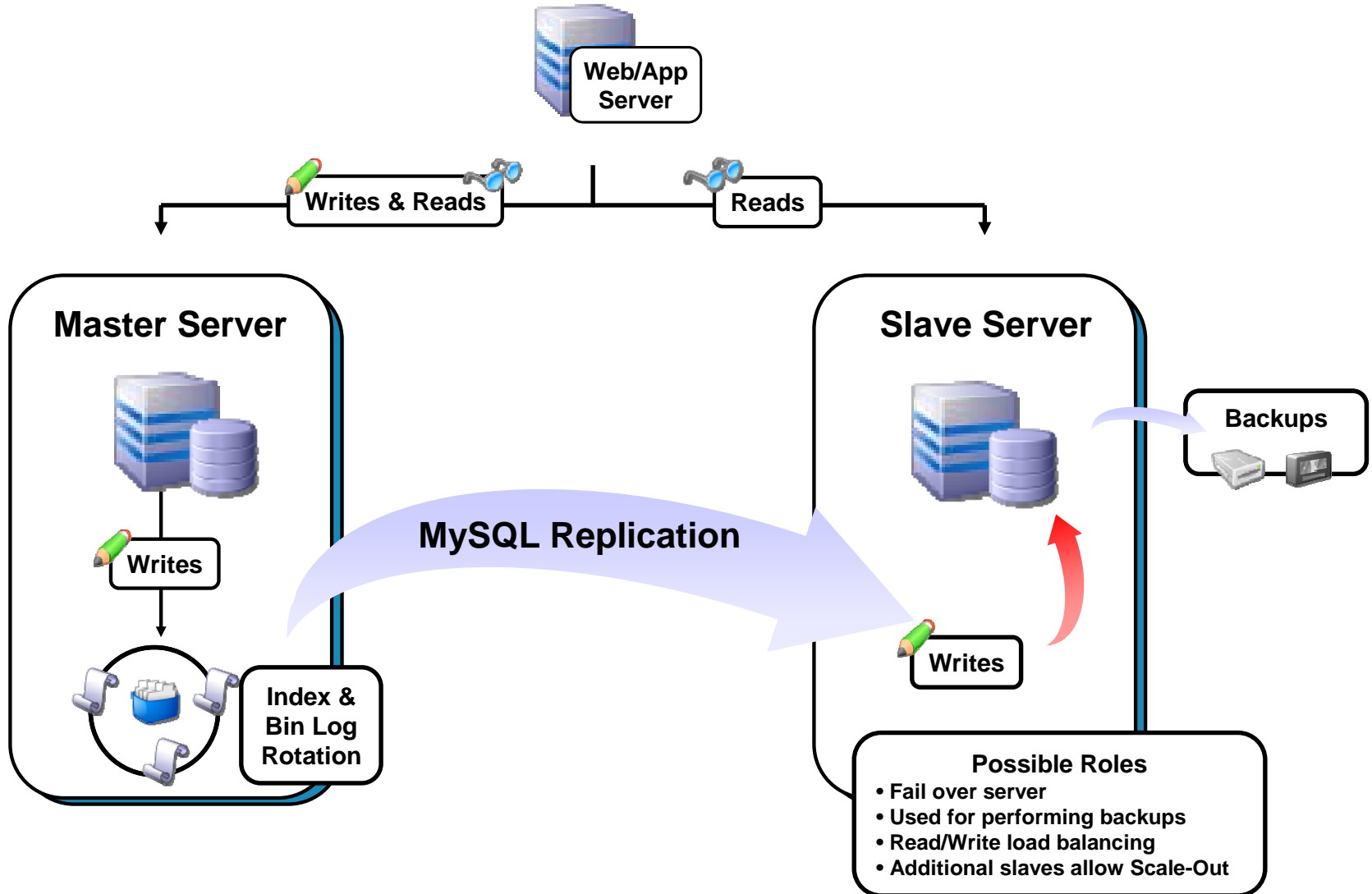
- Horizontal
- Commodity Intel/AMD hardware
- Open source software
- Platform independence
- Add servers to increase capacity & performance



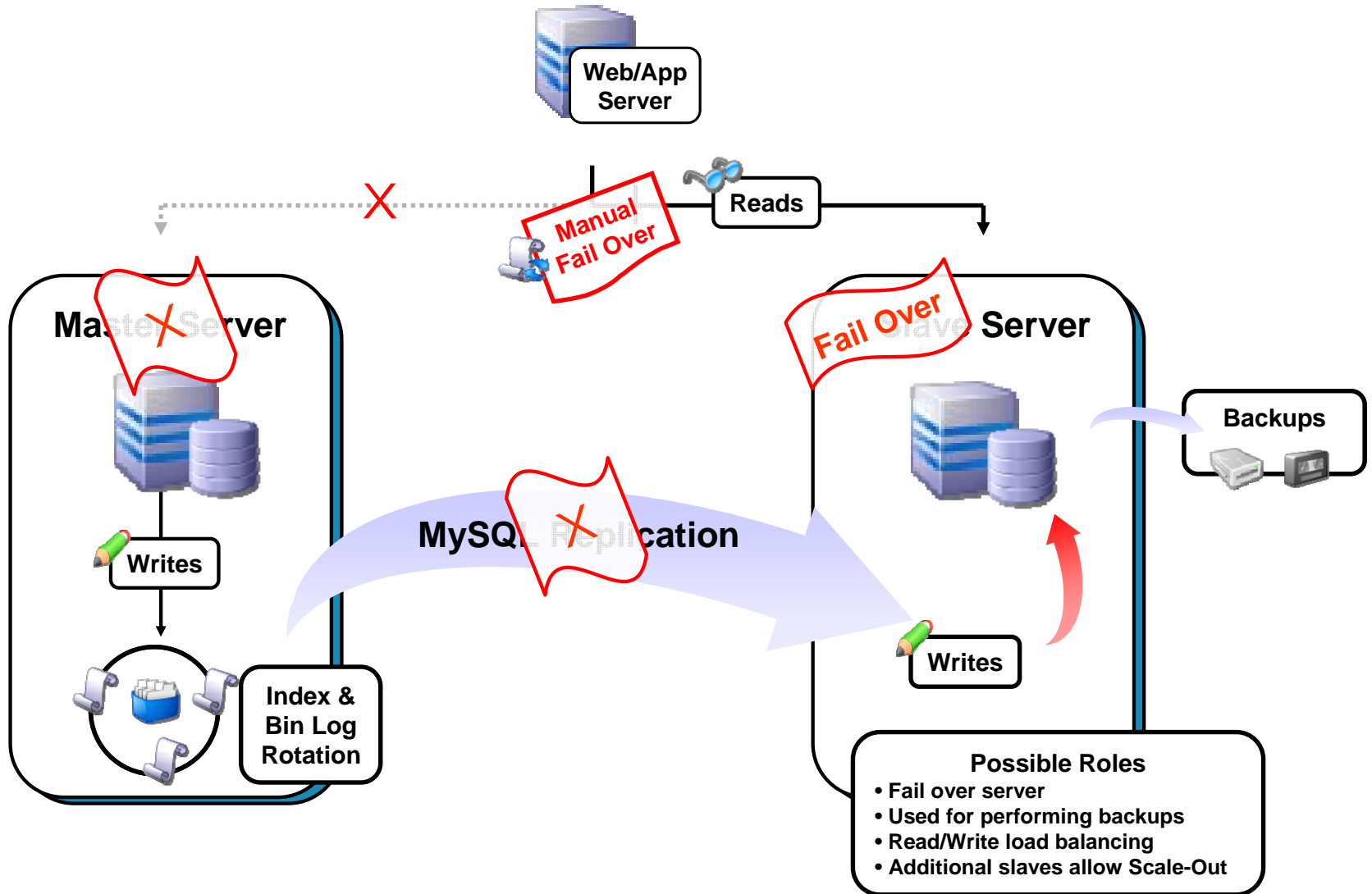
MySQL Replication



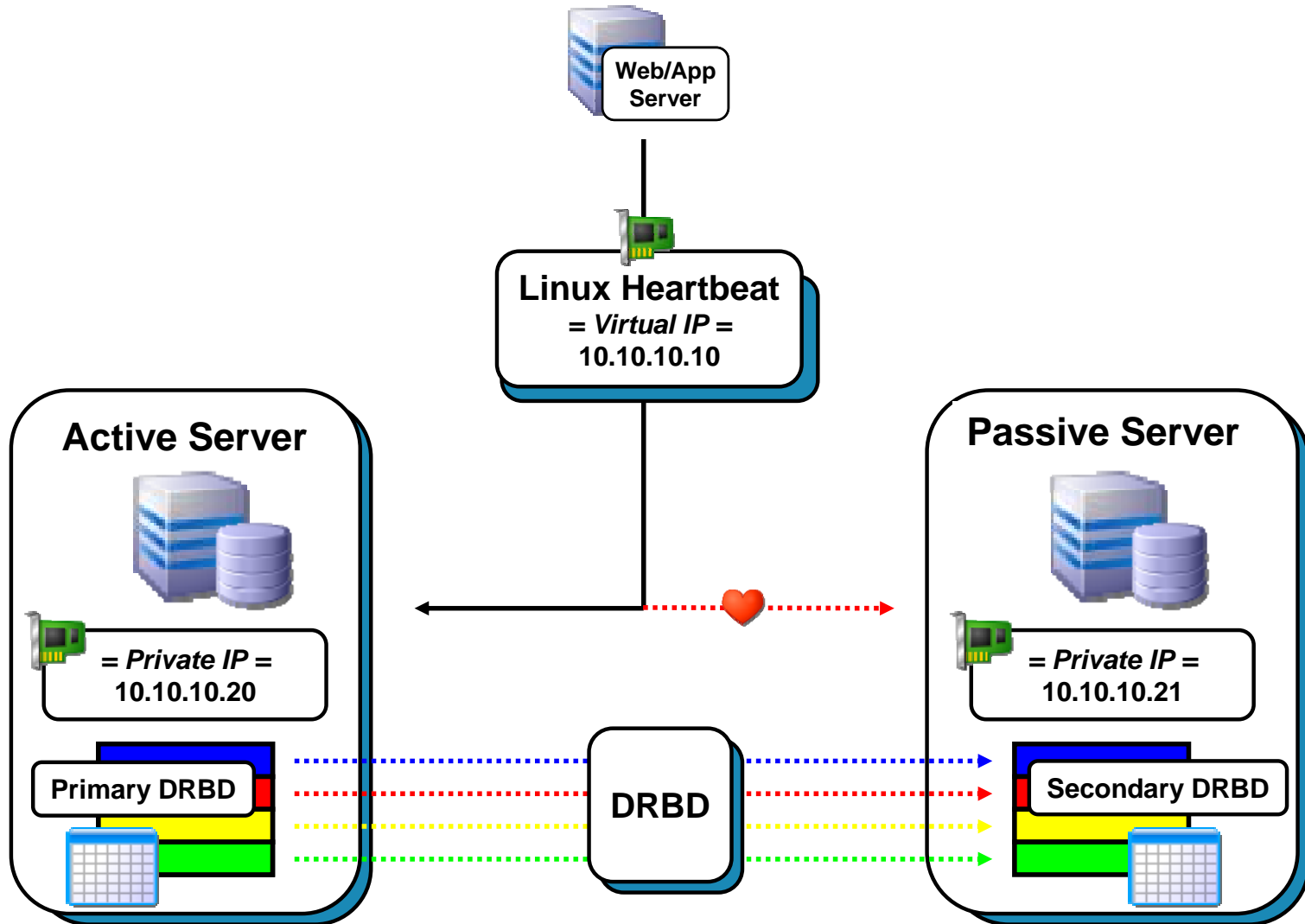
MySQL Replication (Scale-Out)



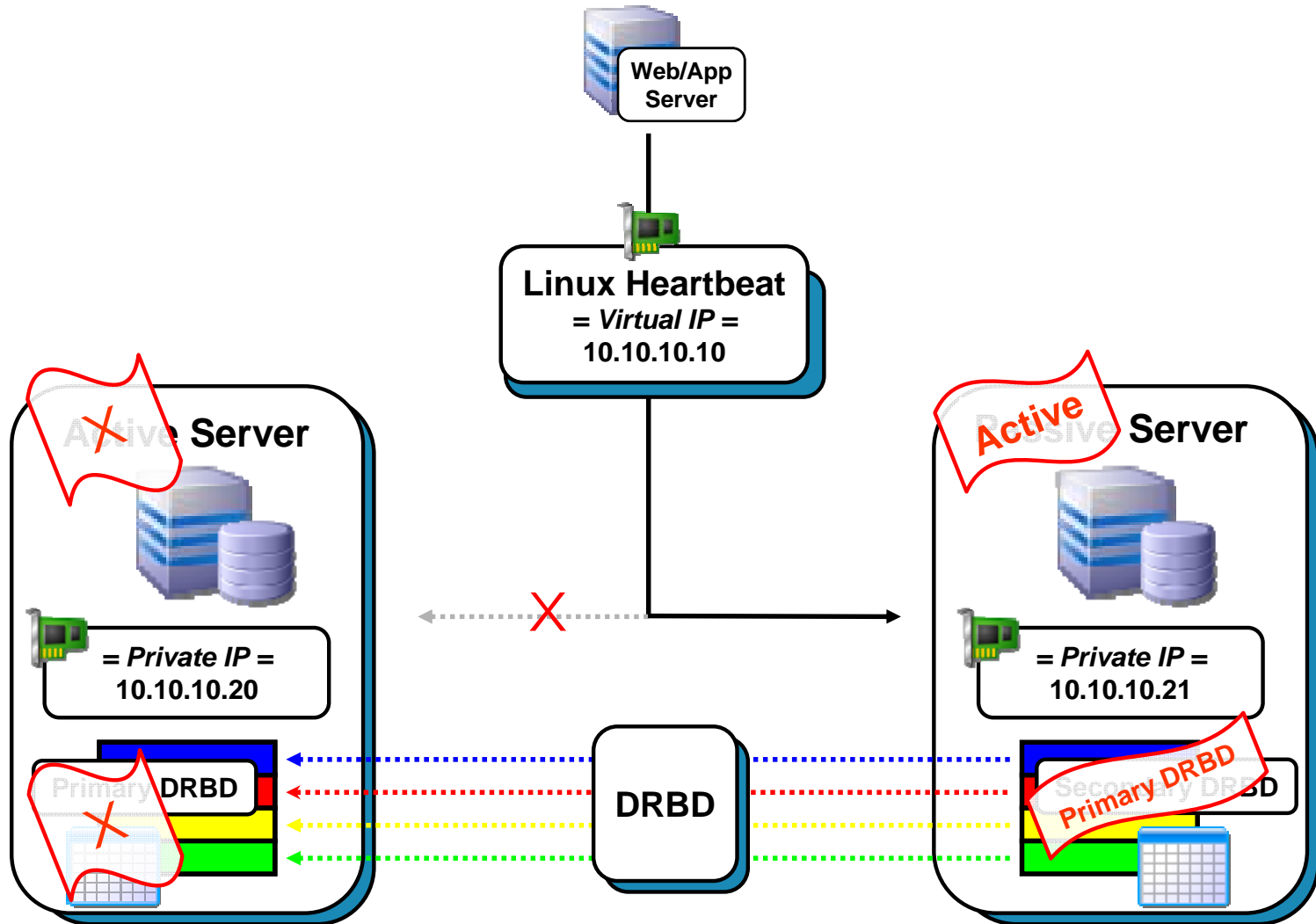
MySQL Replication - Fail Over



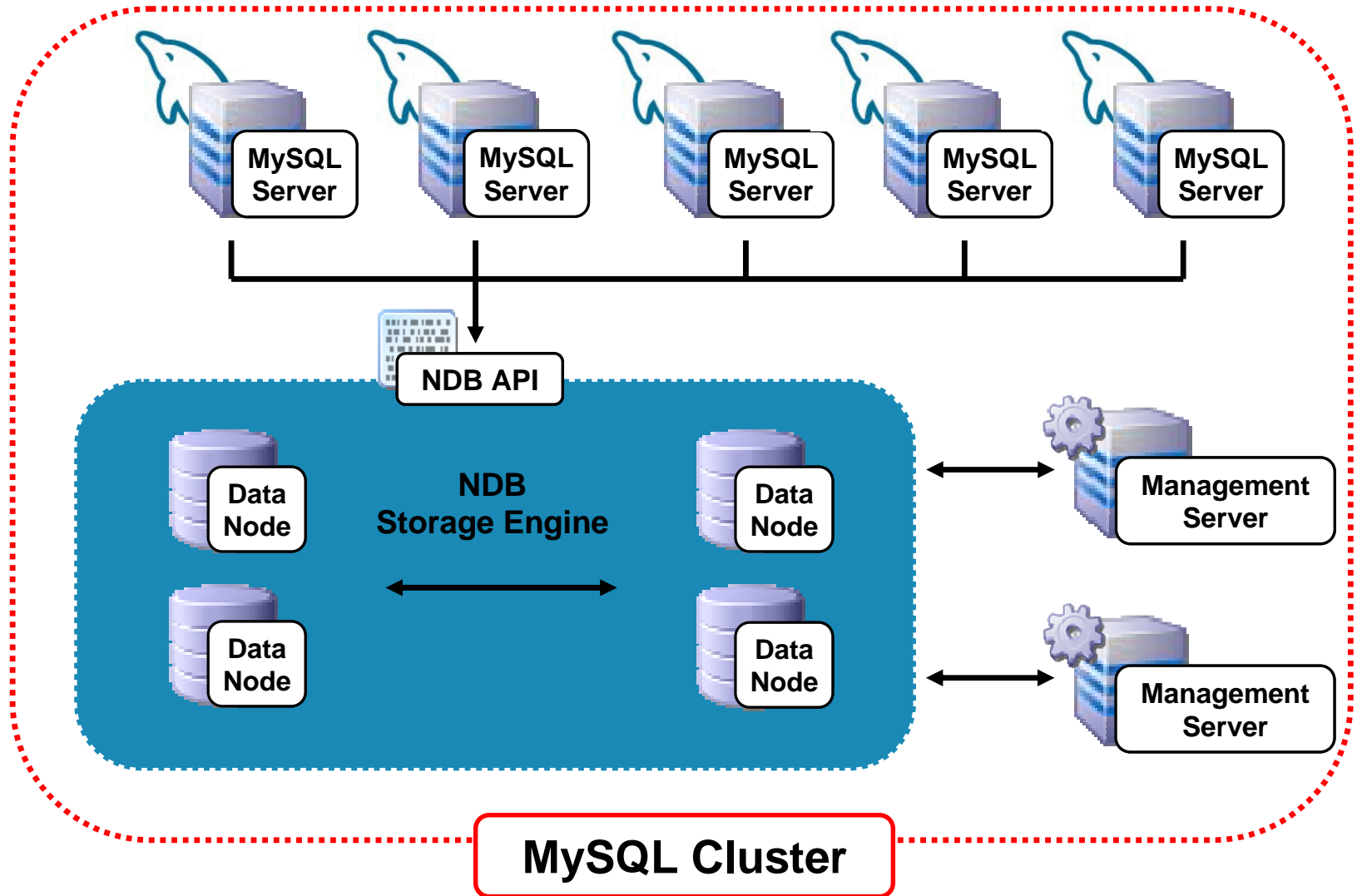
Linux Heartbeat, DRBD & MySQL



Linux Heartbeat, DRBD & MySQL



Scale-Out Cluster Architecture



MySQL Cluster Data Node Architecture

ID	Capital	Country	UTC
1	Copenhagen	Denmark	2
2	Berlin	Germany	2
3	New York City	USA	-5
4	Tokyo	Japan	9
5	Athens	Greece	2
6	Moscow	Russia	4
7	Oslo	Norway	2
8	Beijing	China	8

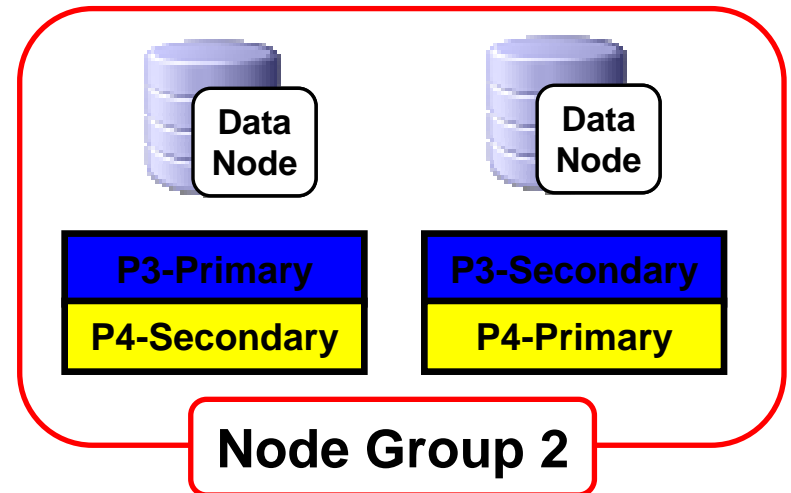
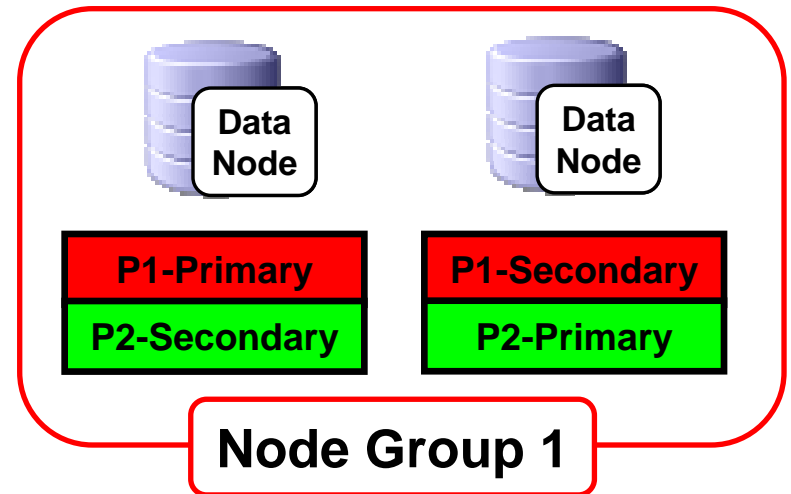
Partition 1

Partition 2

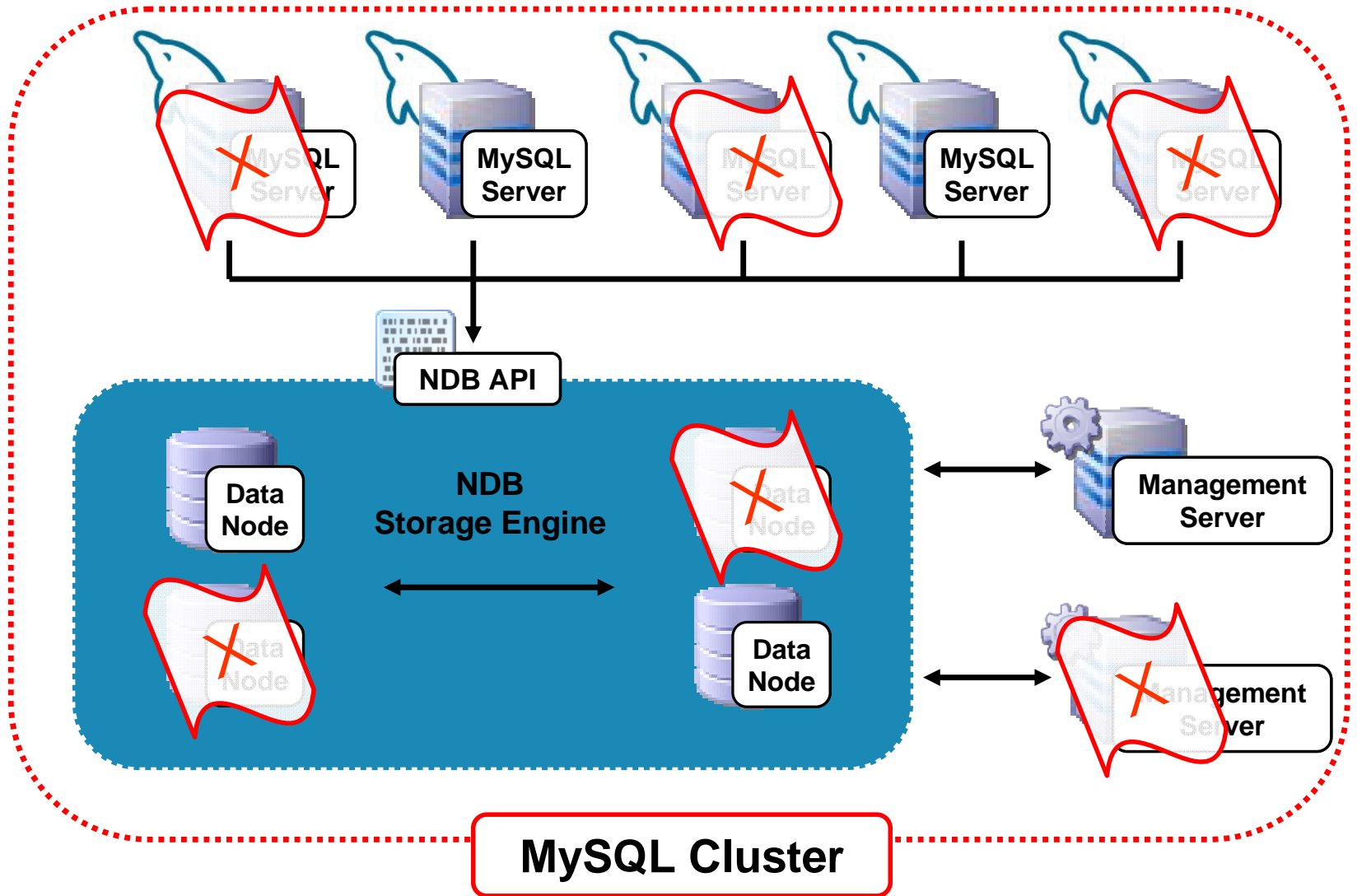
Partition 3

Partition 4

- Four Data Nodes
- Two Replicas
- Two Node Groups



Scale-Out Cluster Architecture – Fail Over



Patterns

- Master/Slave Replication
- Master with Read Only Slaves
- Master to Master
- Ring Replication
- DRDB HA
- DRBD Master Pair with Read Only Slaves
- Master to Master Cross Datacenter + DRDB
- Simple Sharding
- Large Sharing/Scale-out solution + DRDB
- Sharding + DRDB Fail-Over + Geographical Redundancy

Pattern: Master/Slave Replication

Master



Asynchronous



Slave



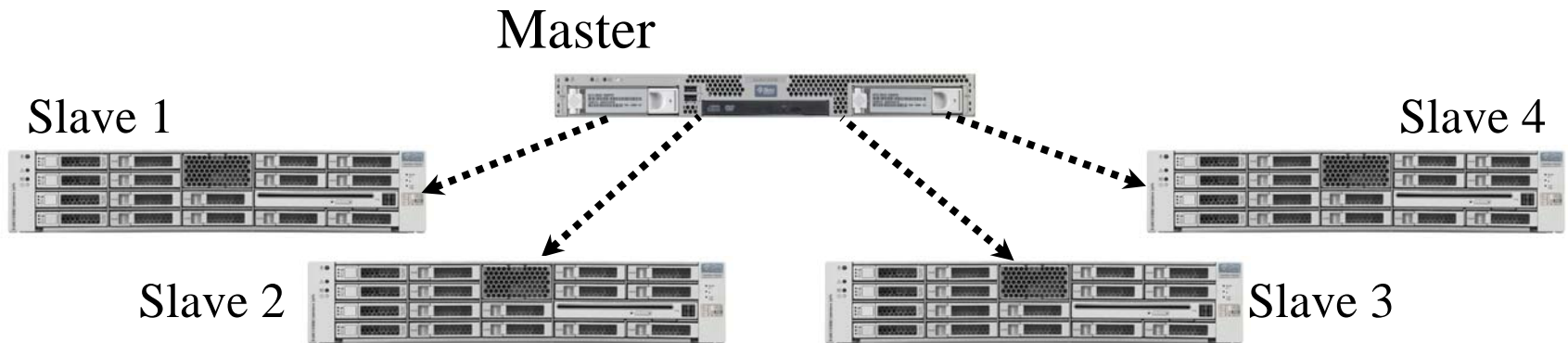
- Advantages

- Can read from either server
- Up to 2x read scaling
- Easy to setup, configure, and maintain
- Easy fail-over, fail-back
- Poor mans HA, ~99.9% (3 nines) HA

- Disadvantages

- ~.5 write scaling or slave will fall behind
- All writes **must** go to the master
- Replication is asynchronous; potential delay in writes to slave
- Data can be left “un-replicated” on the Master in a fail-over.
 - Saving and re-syncing data left in the master on fail-over is a manual task that is not always straight forward.

Pattern: Master with Read Only Slaves

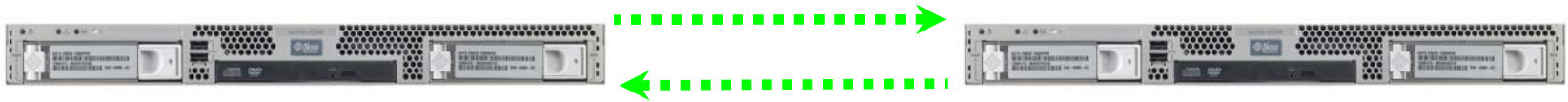


- A master can have many slaves,
 - ~1% load on master per slave
 - 5+ slaves are not unusual, Most I've heard of is 80 slaves
- Reading data is HA, can be designed to be 4-5 nines
- Add more slaves to scale read load
- Can load balance reads across slaves
- Can do long running reports on slave
- Use slave for backup
- JDBC drivers support this architecture

Pattern: Master to Master Replication

Master 1

Master 2



- Advantages
 - Can read from or write to either server
 - Works across WAN - Supports geographical redundancy
 - Can be setup as active/passive
 - Can get some performance advantage by splitting load and keeping it sticky, i.e. east coast always on master 1 and west always on master 2
- Disadvantages
 - Total insert/update/delete load is $\sim .5$ one server or slaves begin falling behind
 - Applications and schema must be designed to eliminate duplicates that would break replication
 - Fail-back can be more complicated

Pattern: Ring Replication



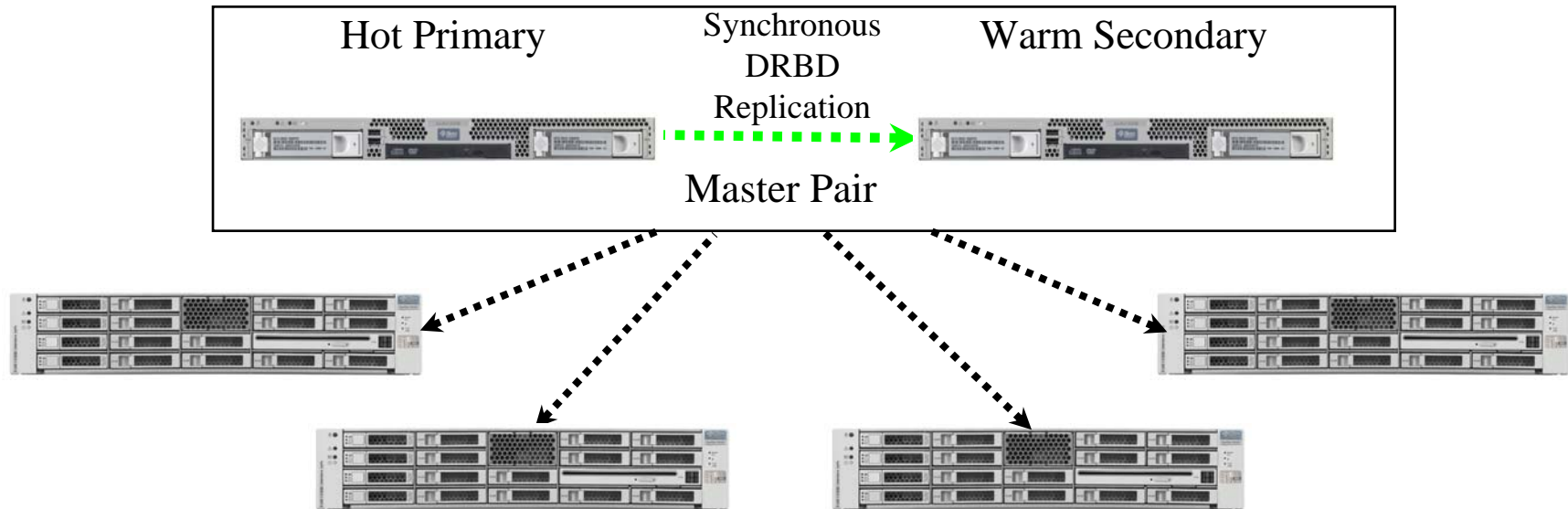
- Advantages
 - Can read from or write any server
 - Works across WAN - Supports multiple geographical redundancy
 - BP - Use DRDB HA for each
- Disadvantages
 - Total insert/update/delete load for **all servers** is ~.5 of typical load of one server or slaves begin falling behind
 - Applications and schema must be designed to eliminate duplicates that would break replication
 - Fail-over and Fail-back is very complicated
- ***Don't try this at home!***
 - ***Unless you really know what you are doing!***

Pattern: DRBD HA



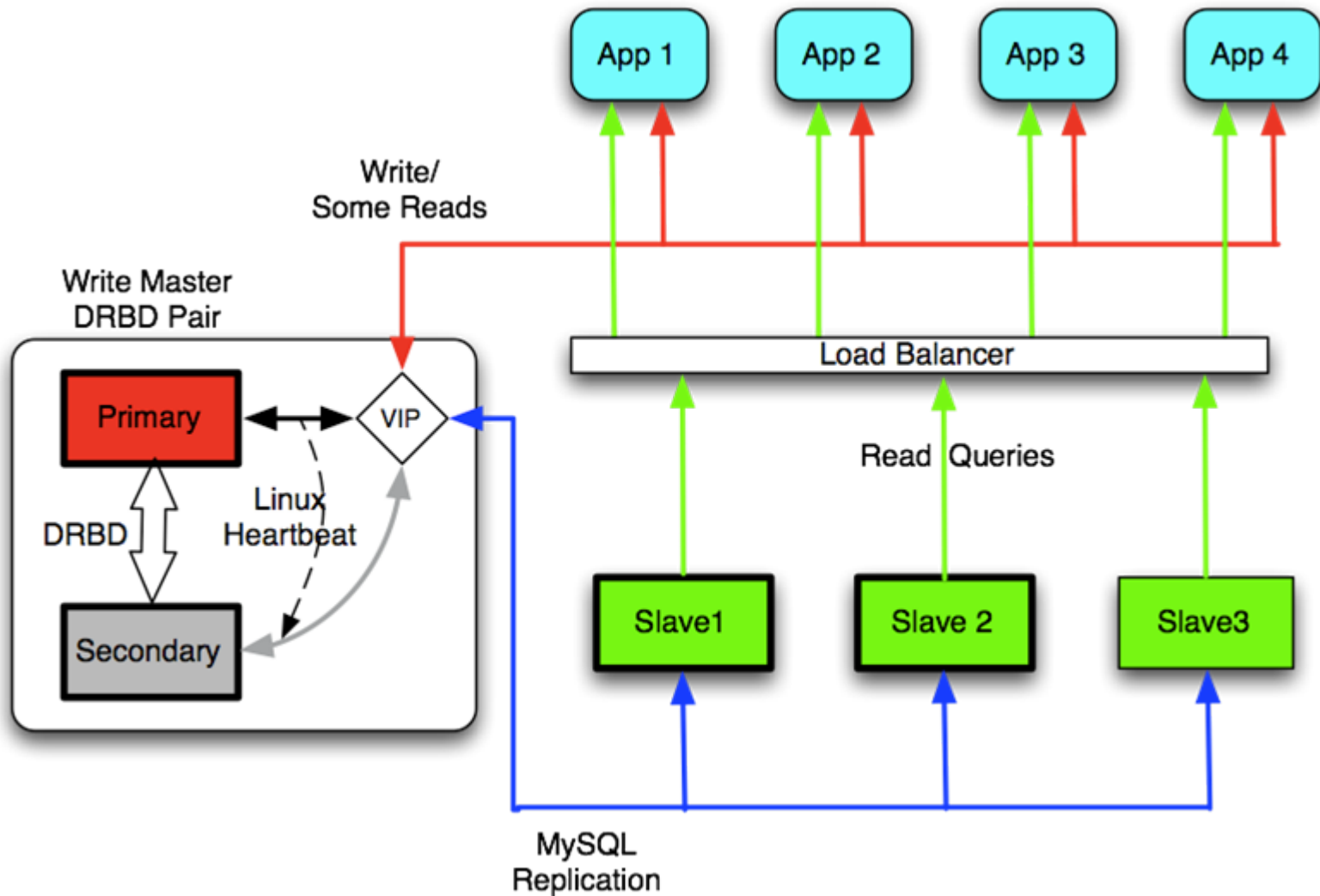
- Advantages
 - No data loss
 - Can replicate from the pair via the fail-over VIP
 - Allows for heavier load on primary than replication on master
 - Fully automated fail-over when using Heartbeat
 - Very easy to rebuild old primary as new secondary after fail-over
 - Can be 3-4 9s Available
- Disadvantages
 - Secondary is passive
 - 30-300 Second delay in recovery after fail-over
 - Only works on Linux

Pattern: DRBD Master Pair with Read Only Slaves

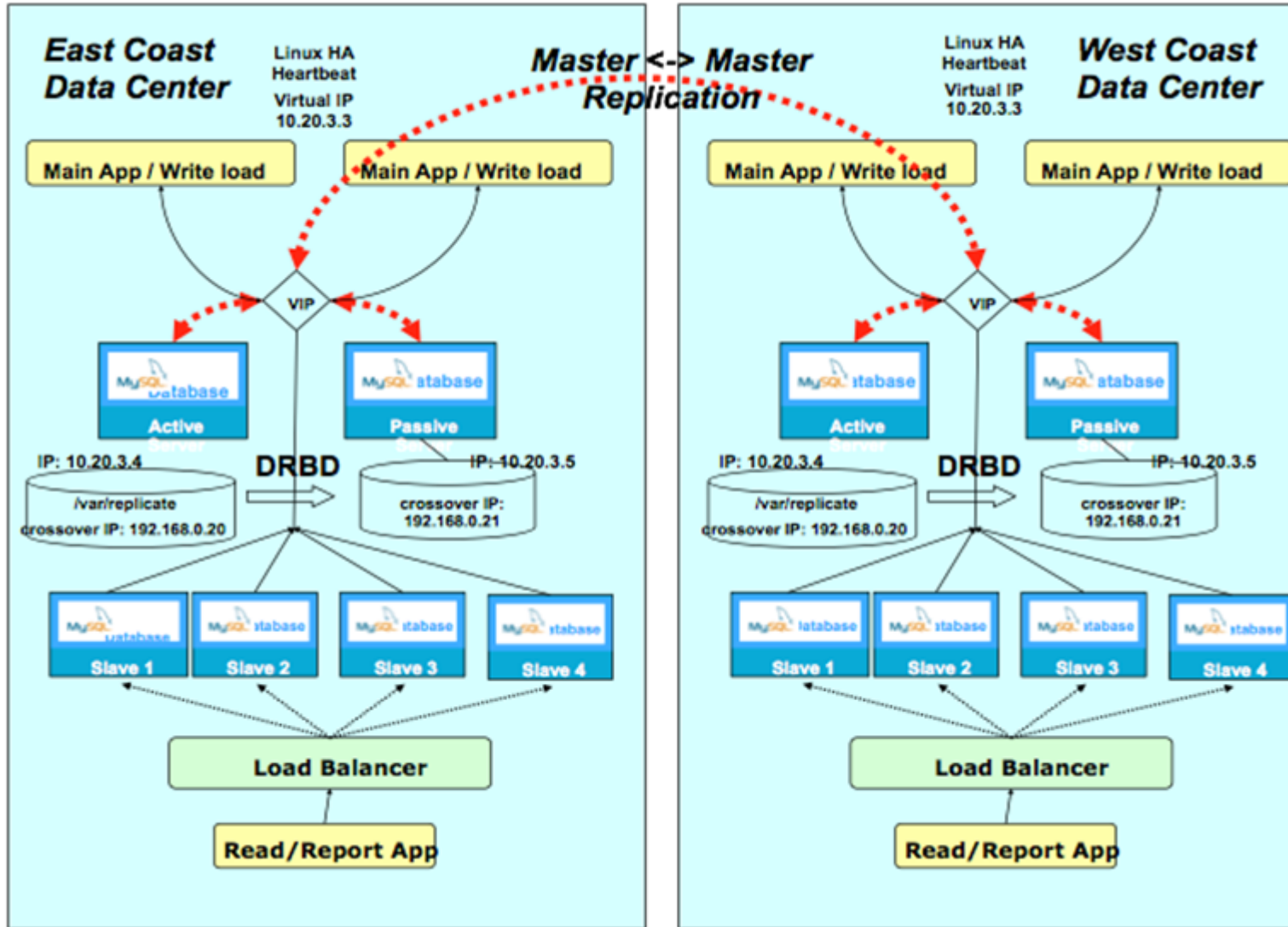


- HA - Both for reads and write
- Scale-out for reads
- Often the optimal solution for most needs
- Replication is via fail-over VIP so slaves transparently fail-over to secondary on primary failure
- Can load balance across slaves

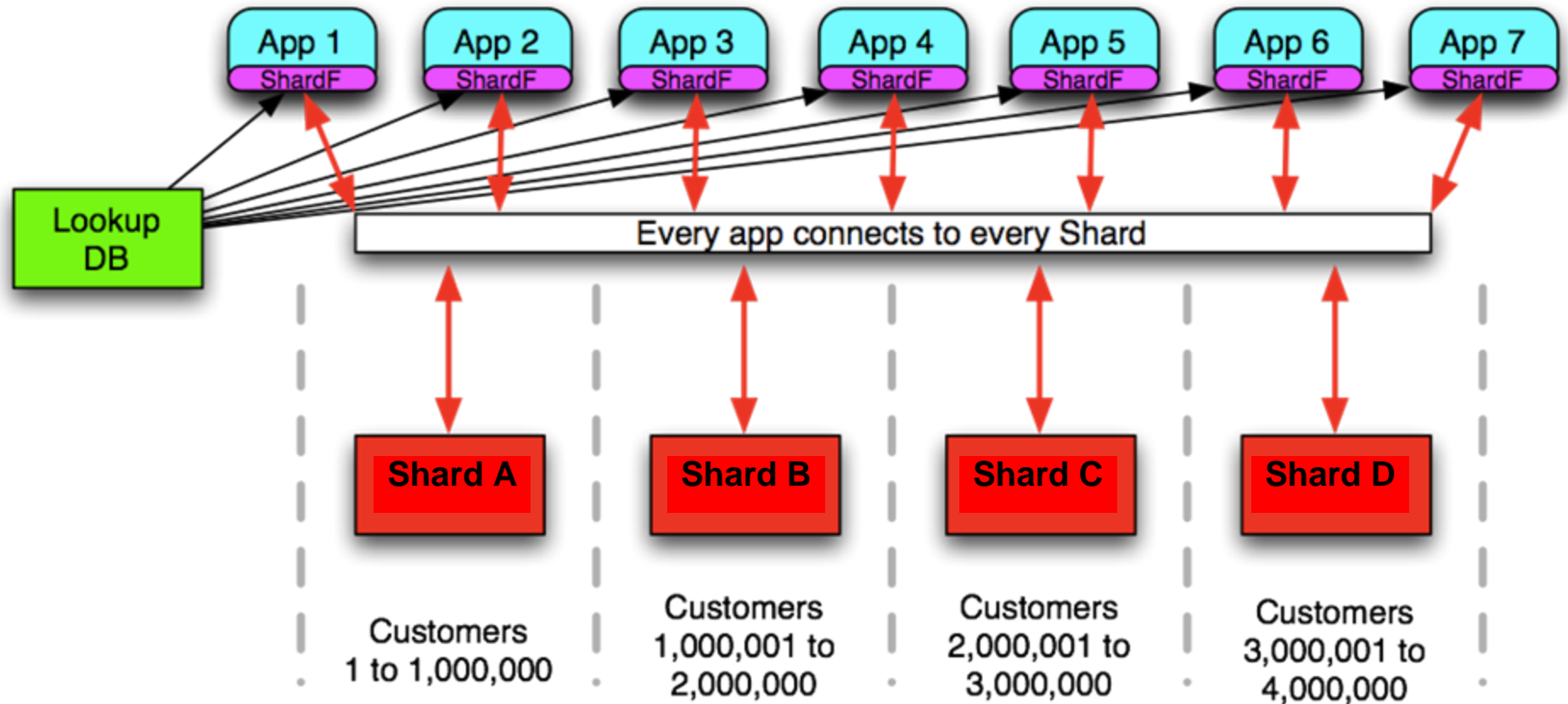
Pattern: DRBD Master Pair with Read Only Slaves



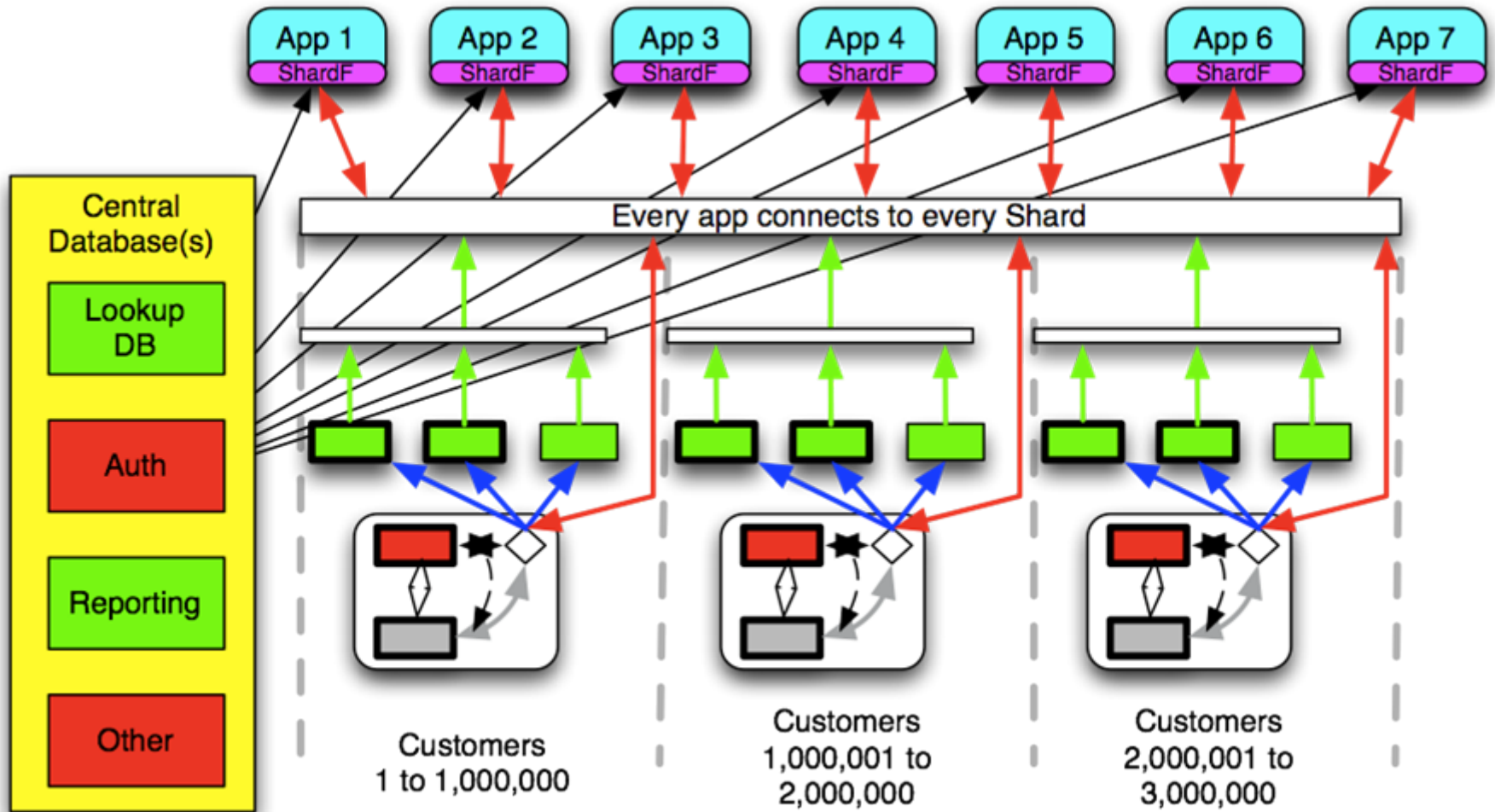
Pattern: Master to Master Cross Datacenter + DRDB



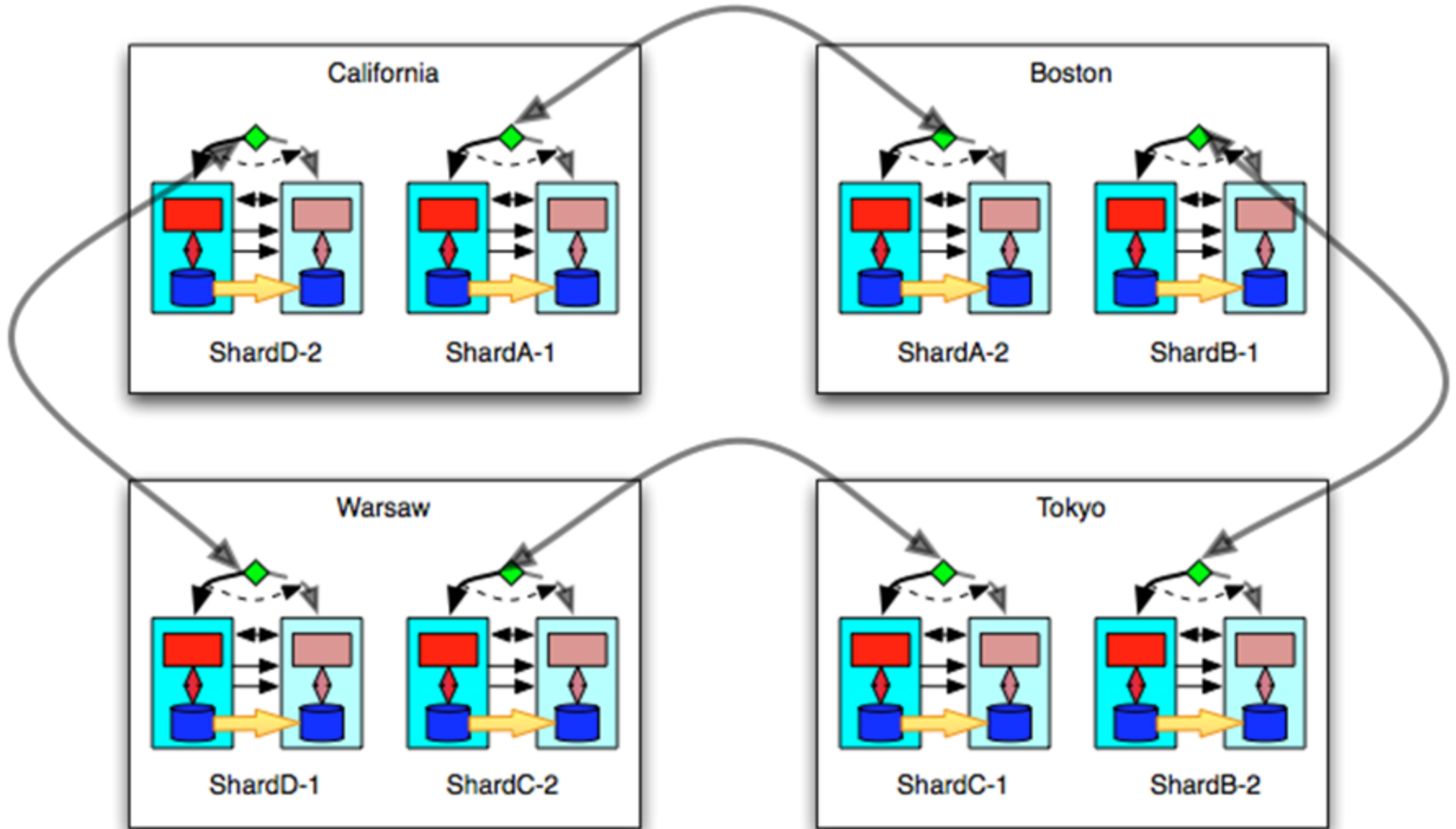
Pattern: Simple Sharding

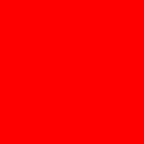


Pattern: Large Sharing/Scale-out solution + DRDB



Pattern: Sharding + DRDB Fail-Over + Geographical Redundancy





The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.

The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.