

A State of Multi-Master Replication

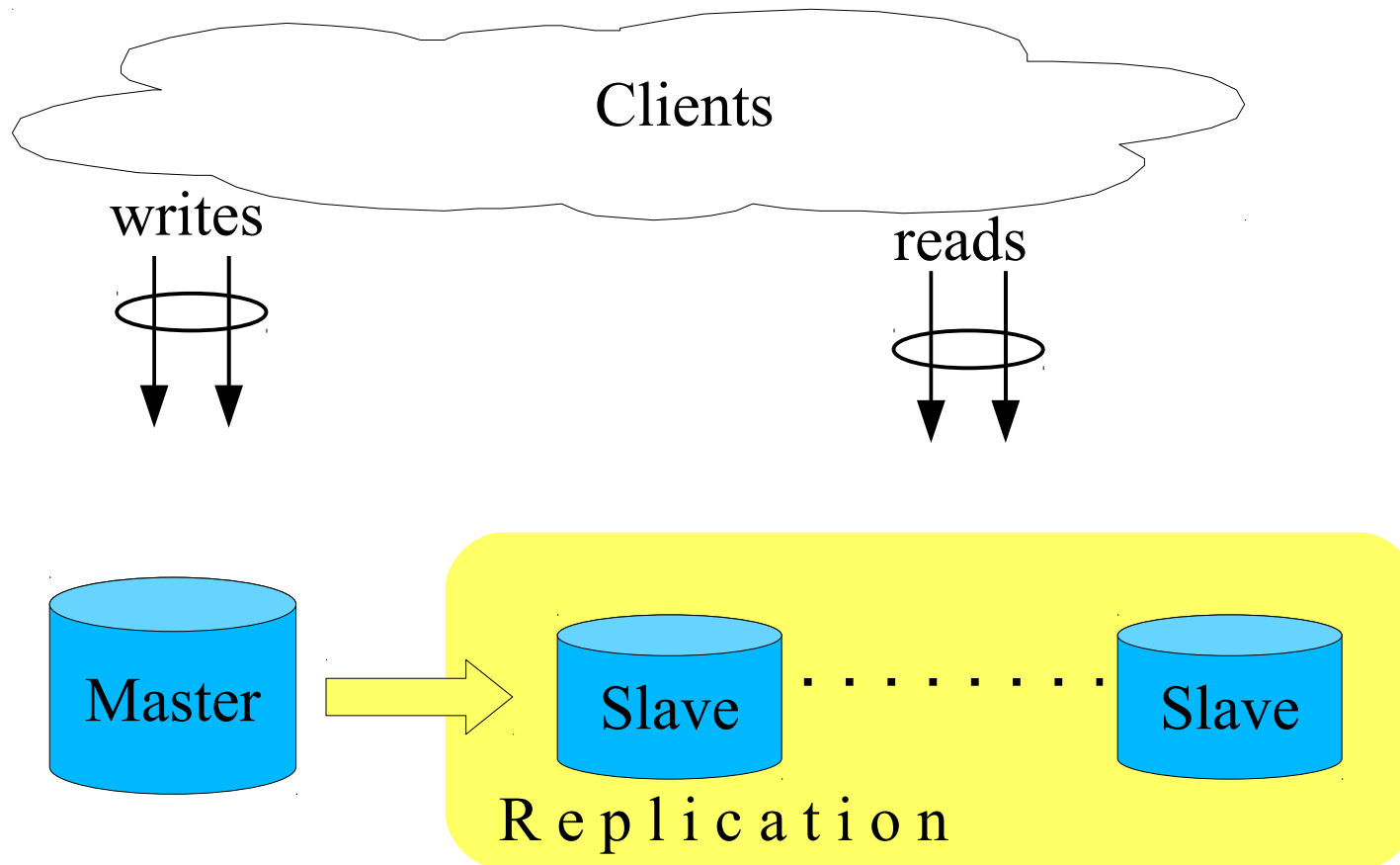
Seppo Jaakola
Alexey Yurchenko

Codership

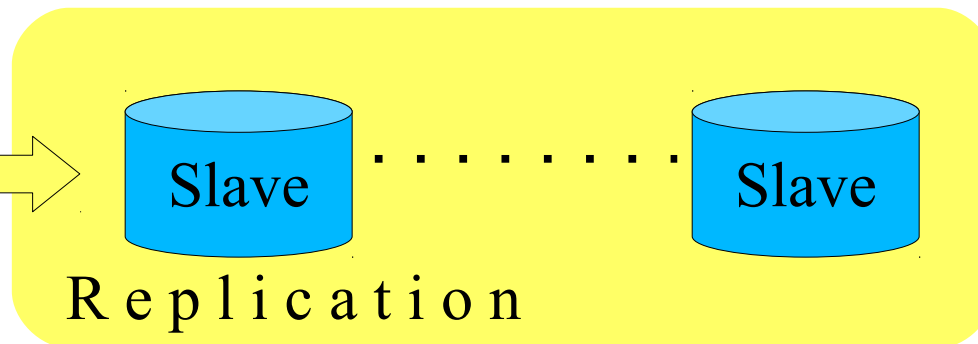
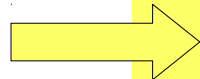
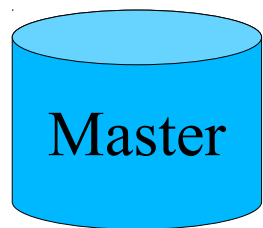
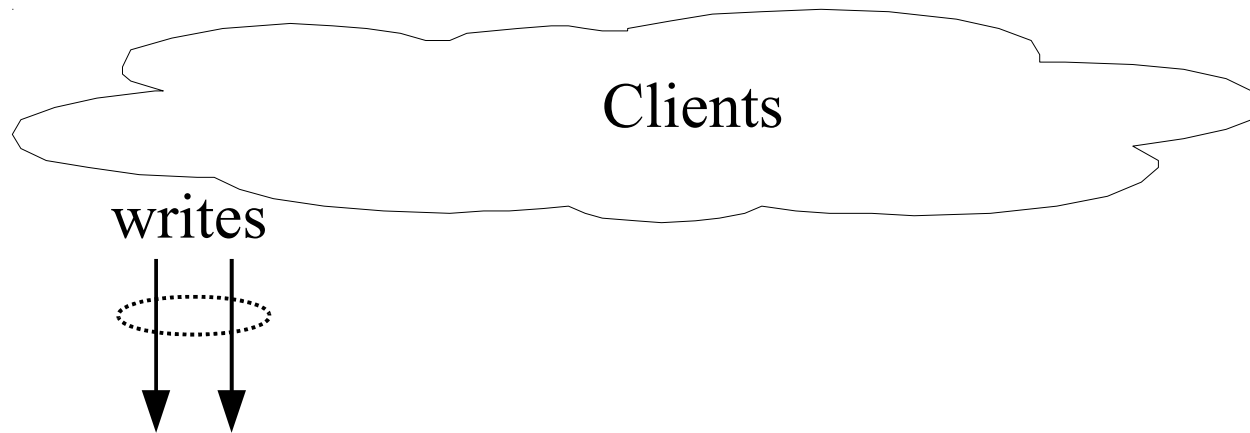
Contents

1. Master/Slave vs Multi-Master
2. Multi-Master Architectures
3. Multi-Master Use Cases
4. Solutions Out There
5. Summary

Master / Slave Replication



Master / Slave Replication

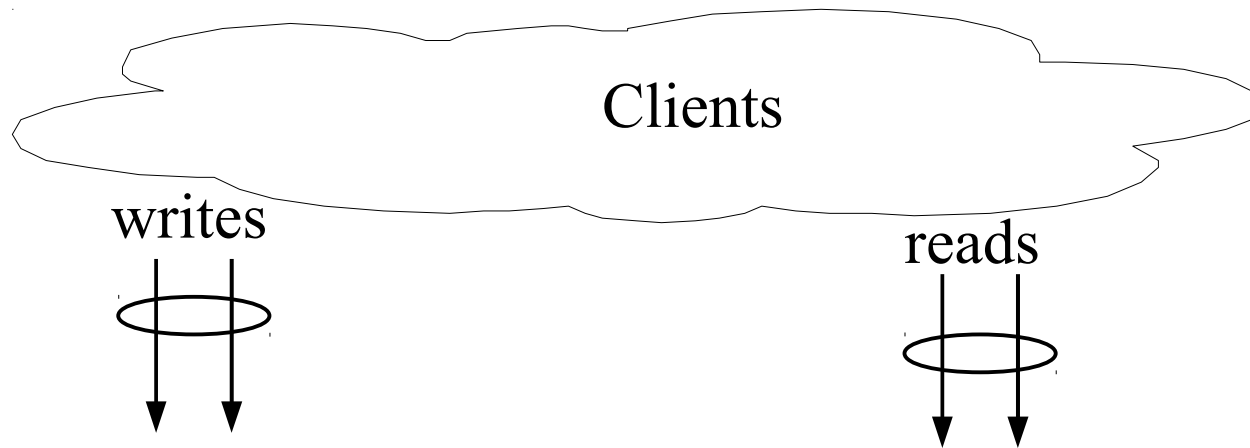


Issues:



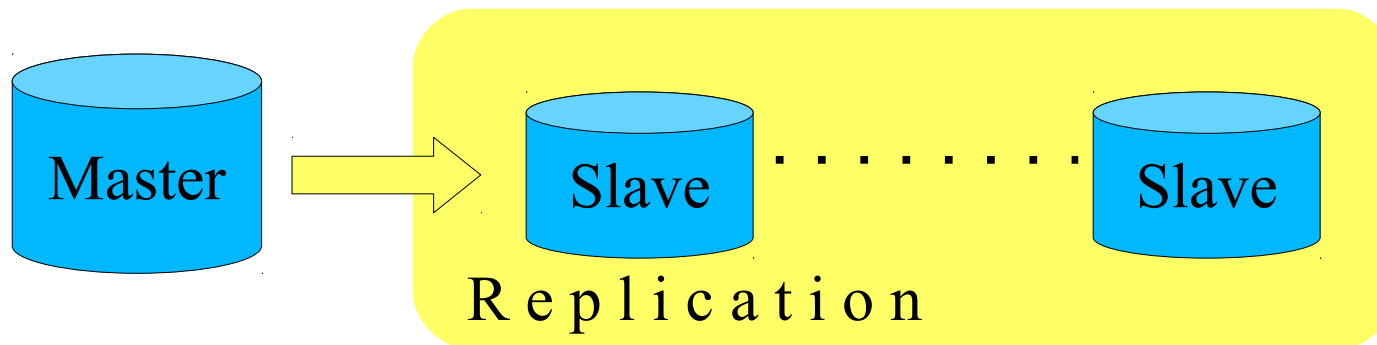
Locating the
Master

Master / Slave Replication

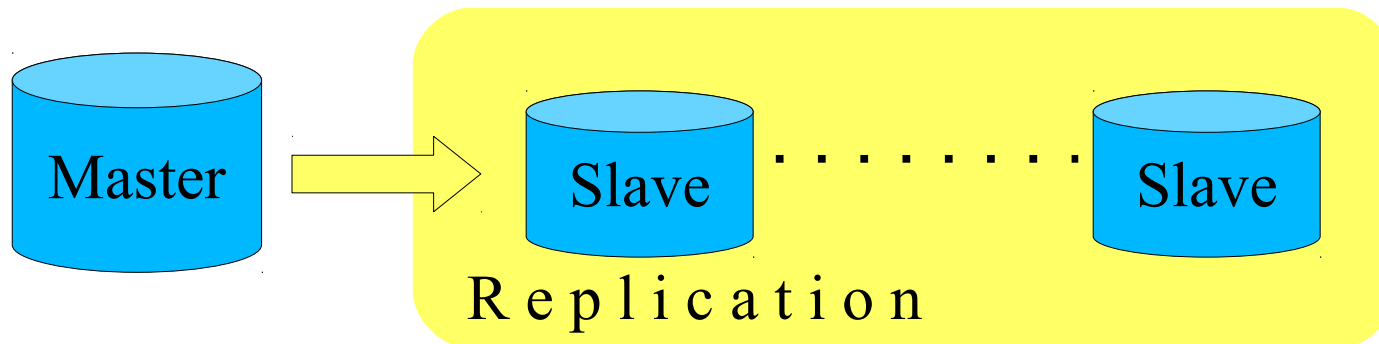
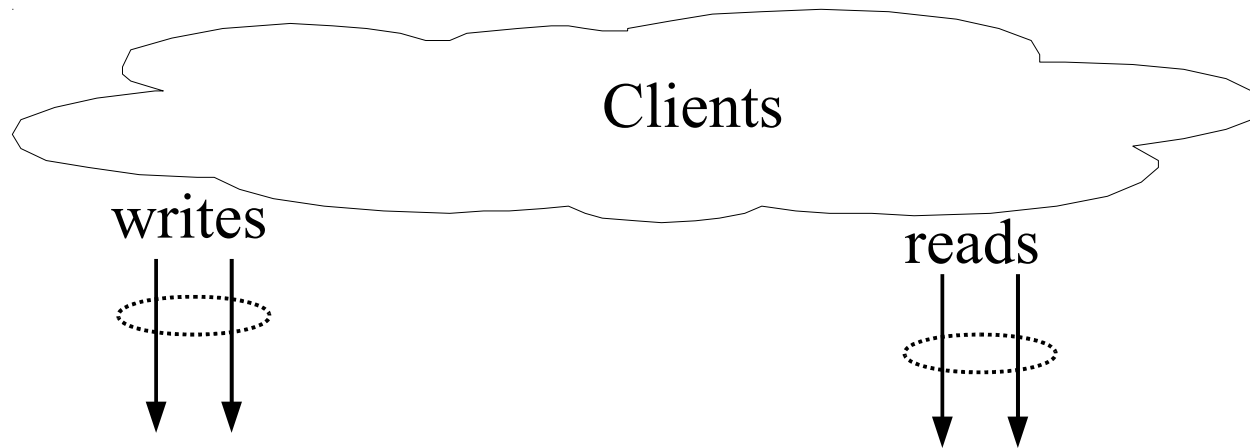


Issues:

- ← Locating the Master
- ← read / write splitting



Master / Slave Replication



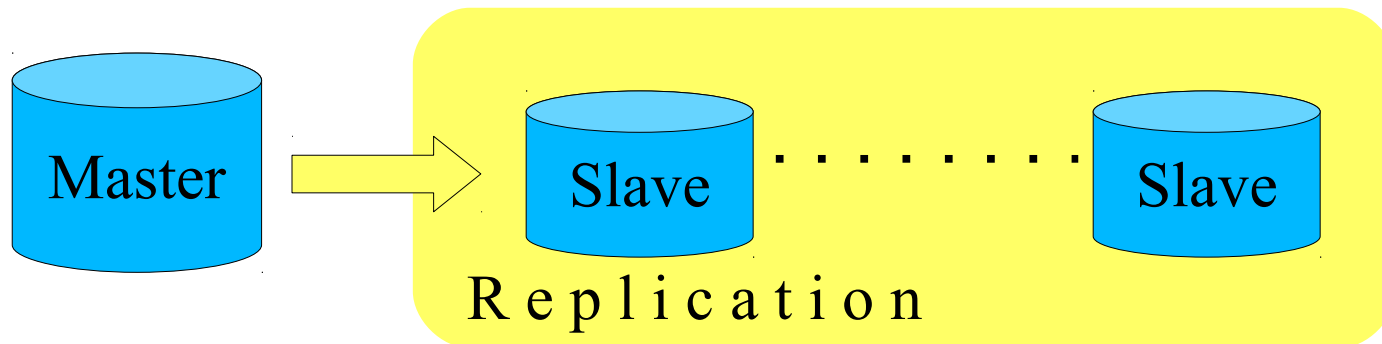
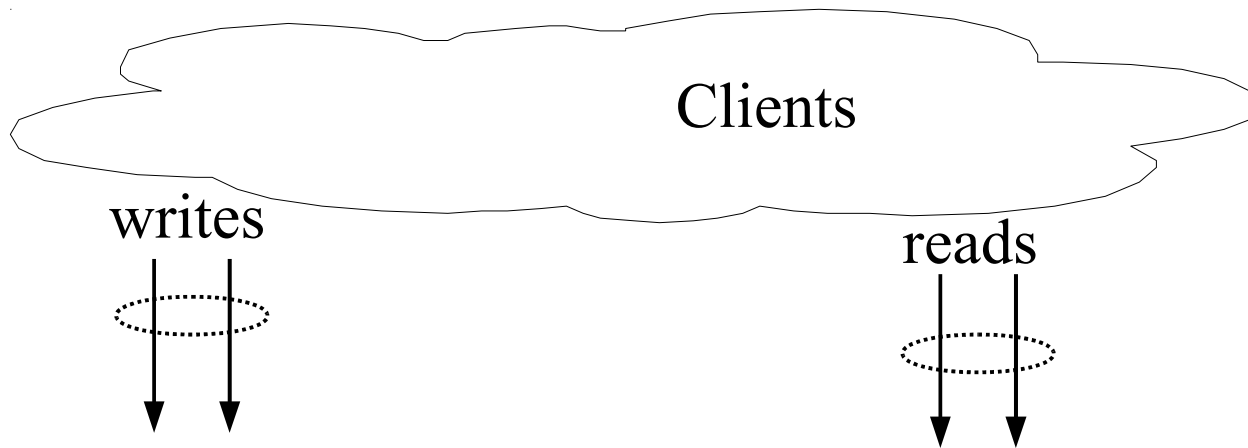
Issues:

← Locating the Master

← read / write splitting

← Slave lag / Consistency

Master / Slave Replication



Issues:

← Locating the Master

← read / write splitting

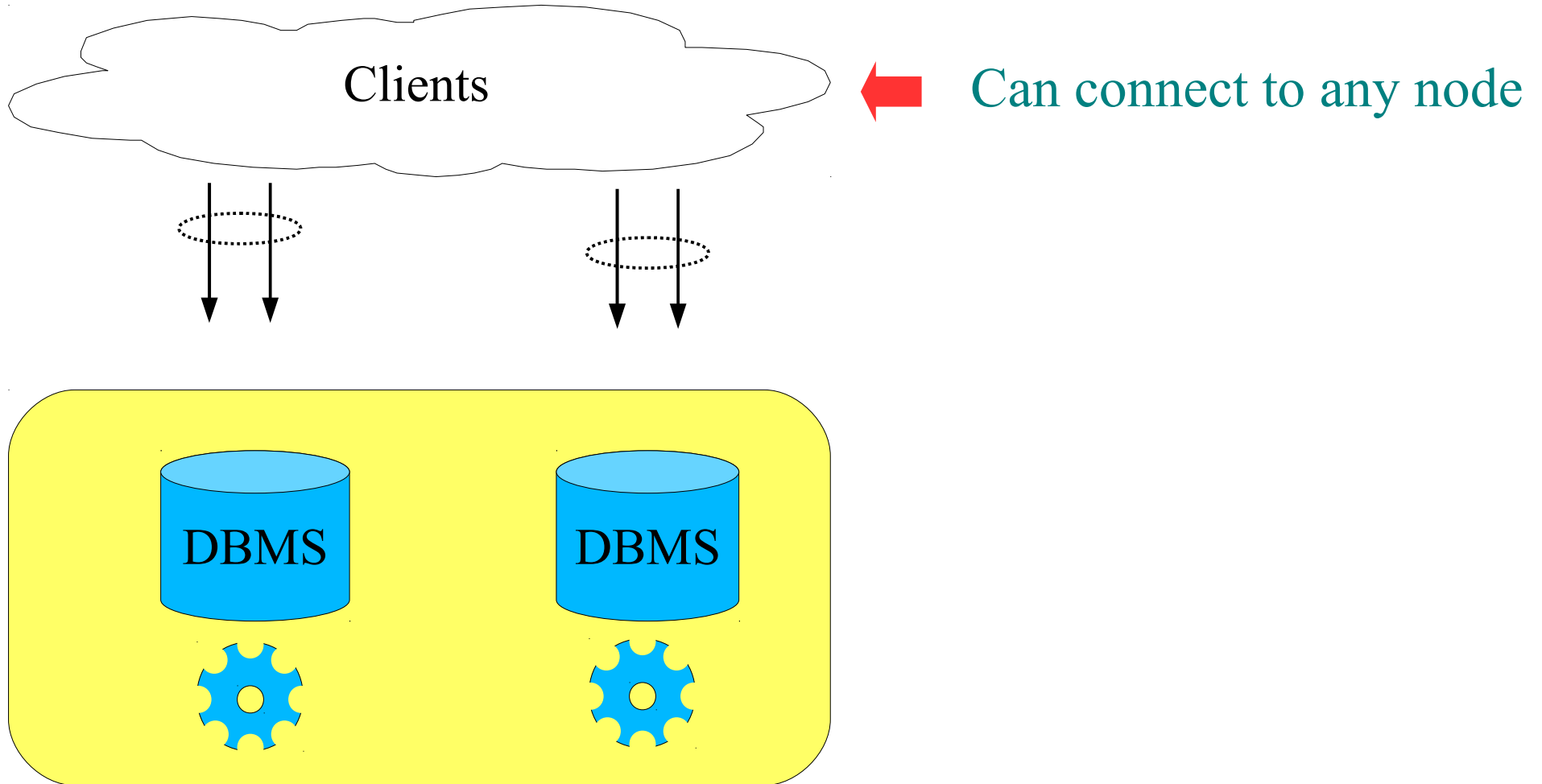
← Slave lag / Consistency

← Master failover

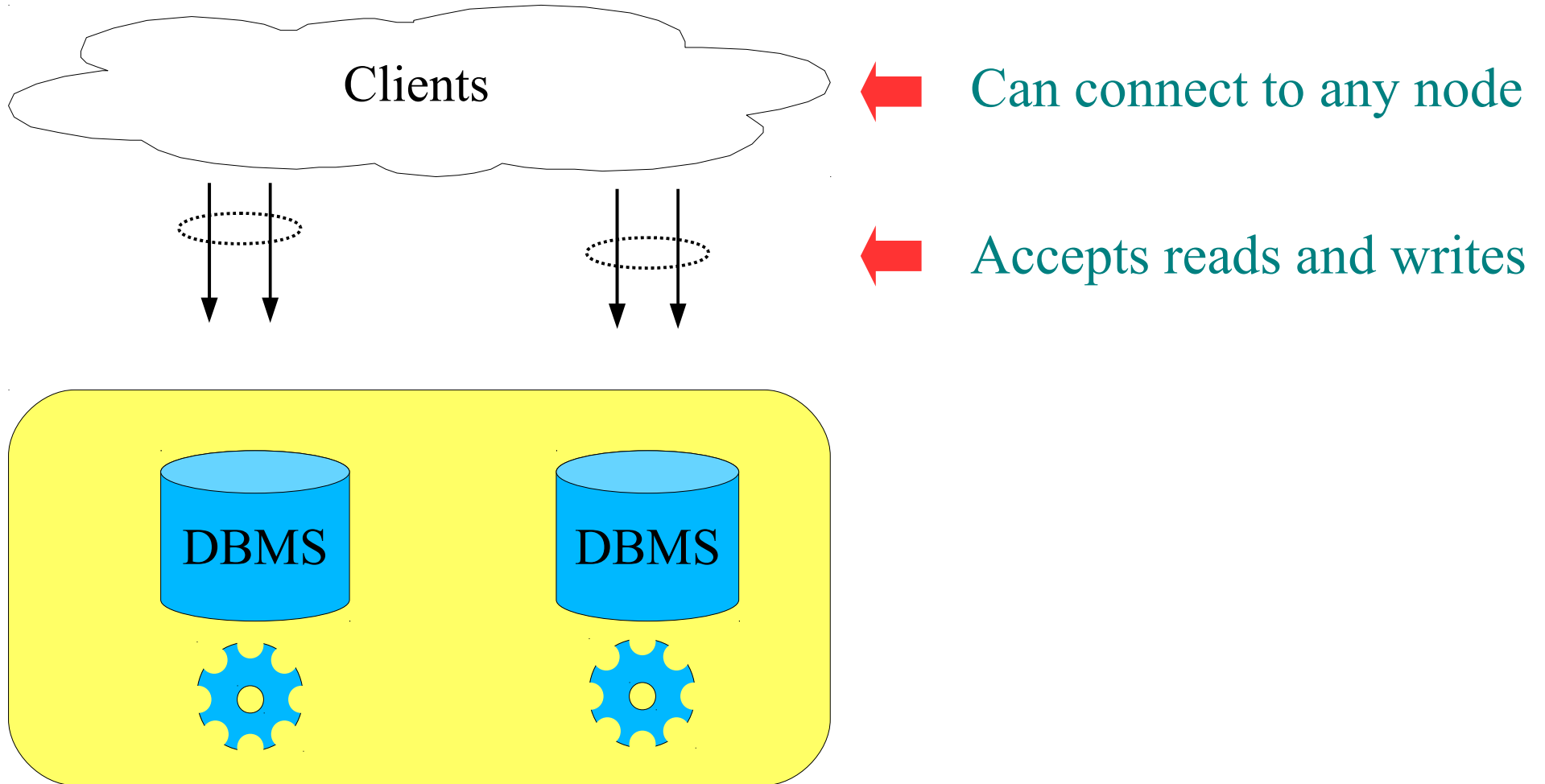
Why Multi-Master?

- Simple to use, less responsibilities in application side
- Local access in WAN cluster
- Write Scalability

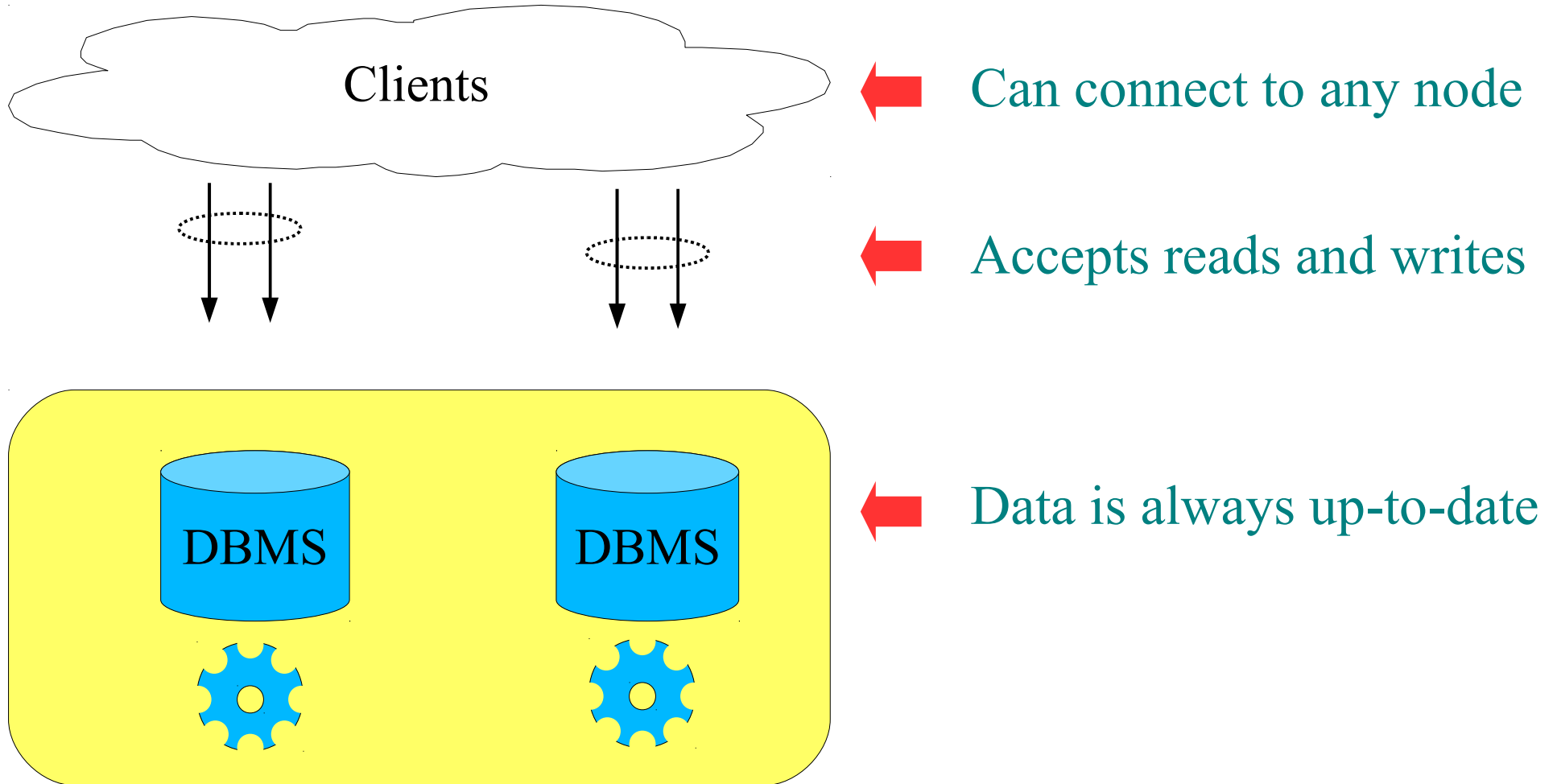
Multi-Master Topology



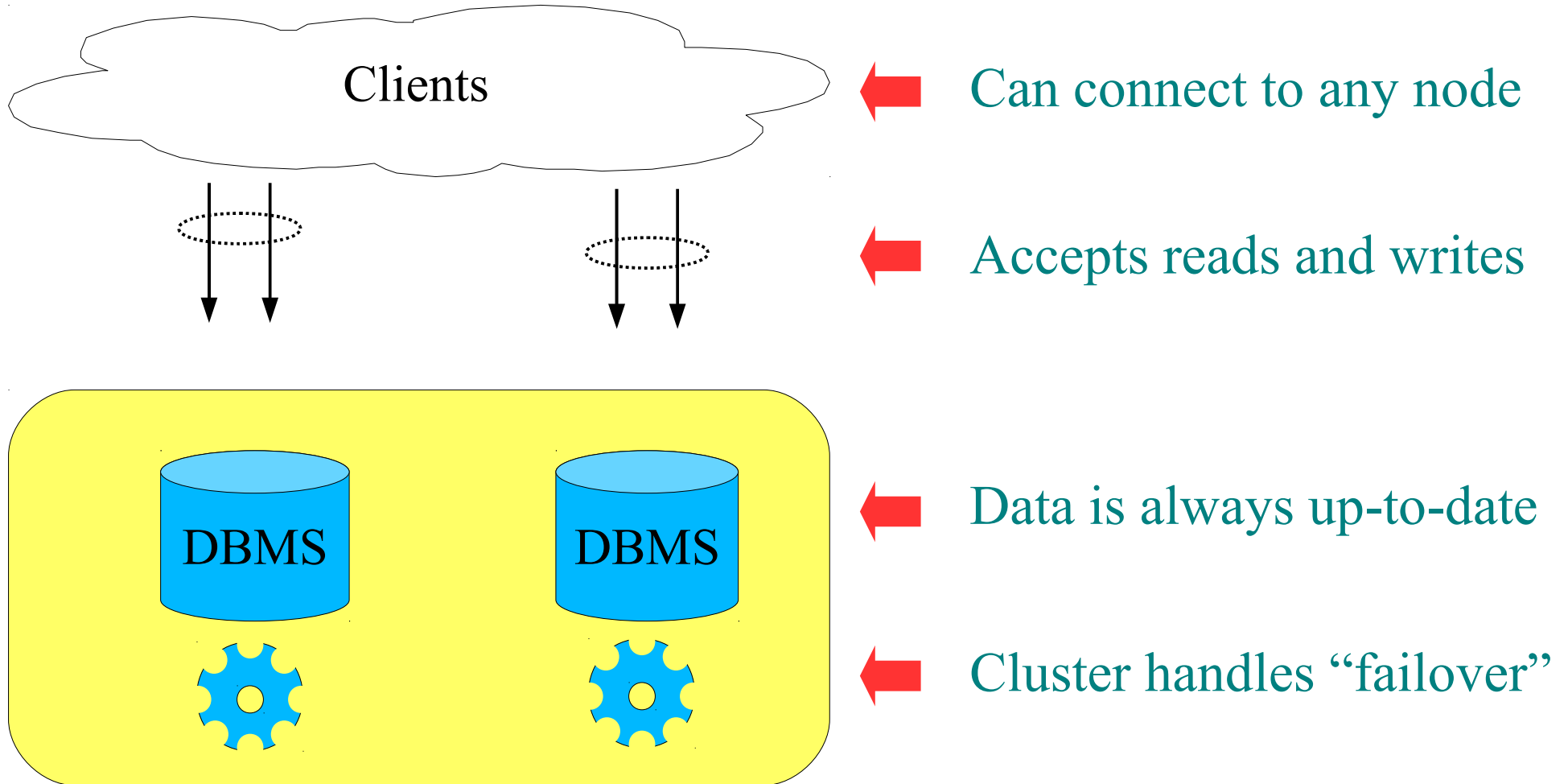
Multi-Master Topology



Multi-Master Topology



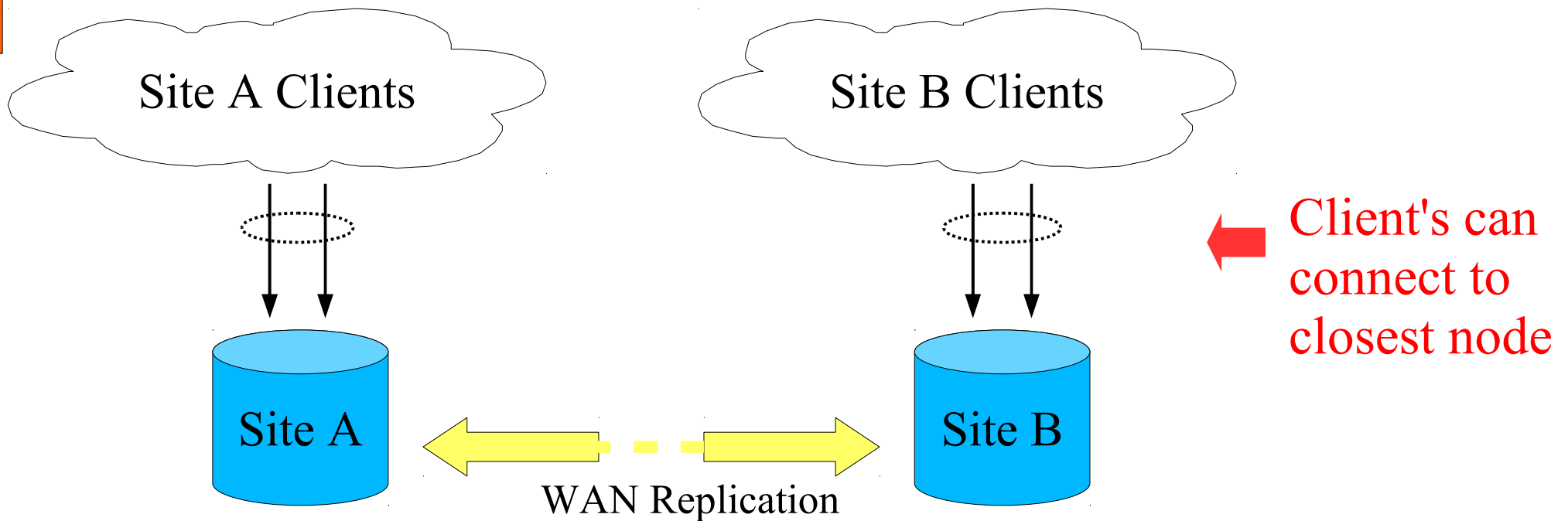
Multi-Master Topology



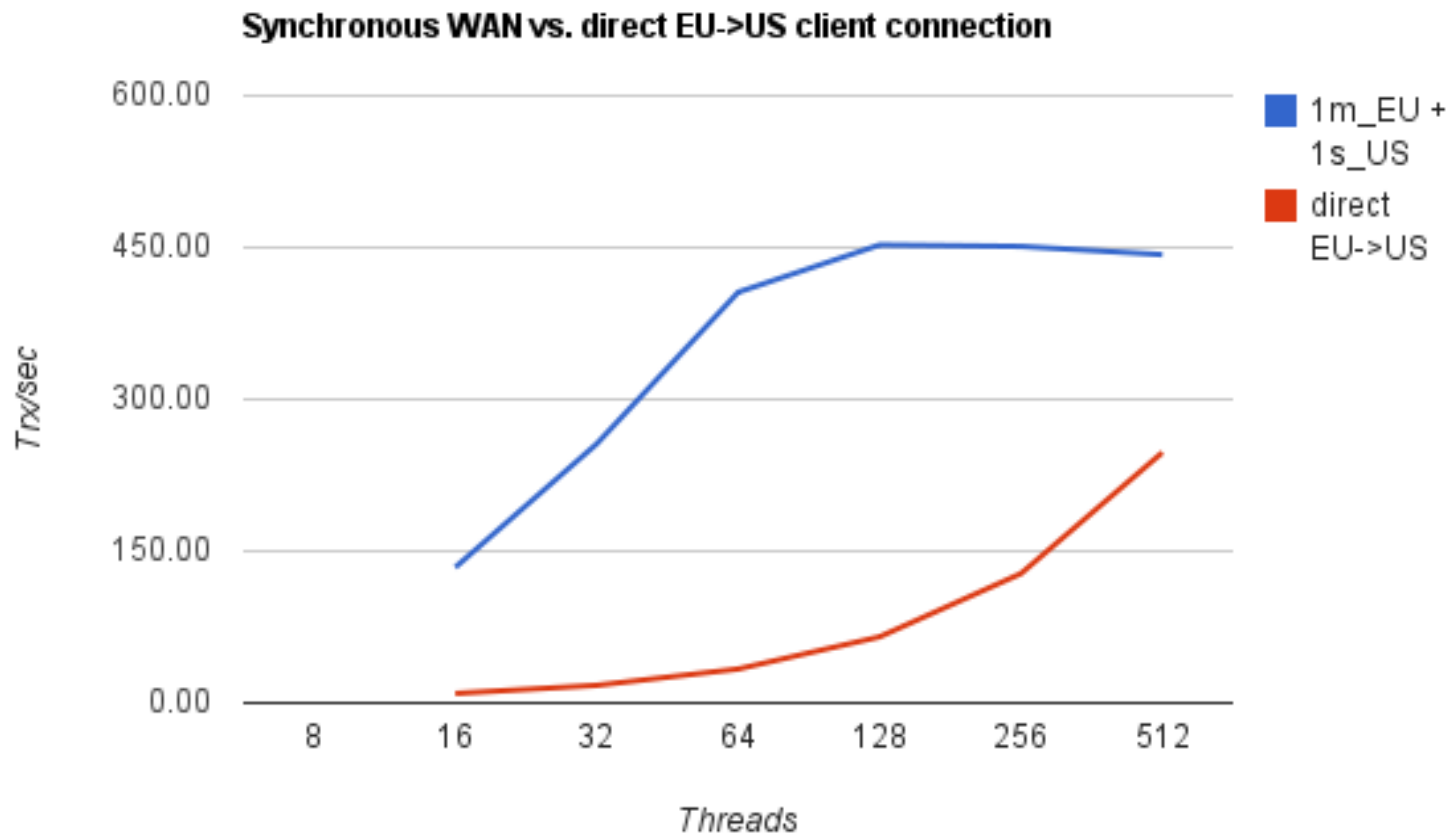
Why Multi-Master?

- Simple to use, less responsibilities in application side
- Local access in WAN cluster
- Write Scalability

WAN Replication

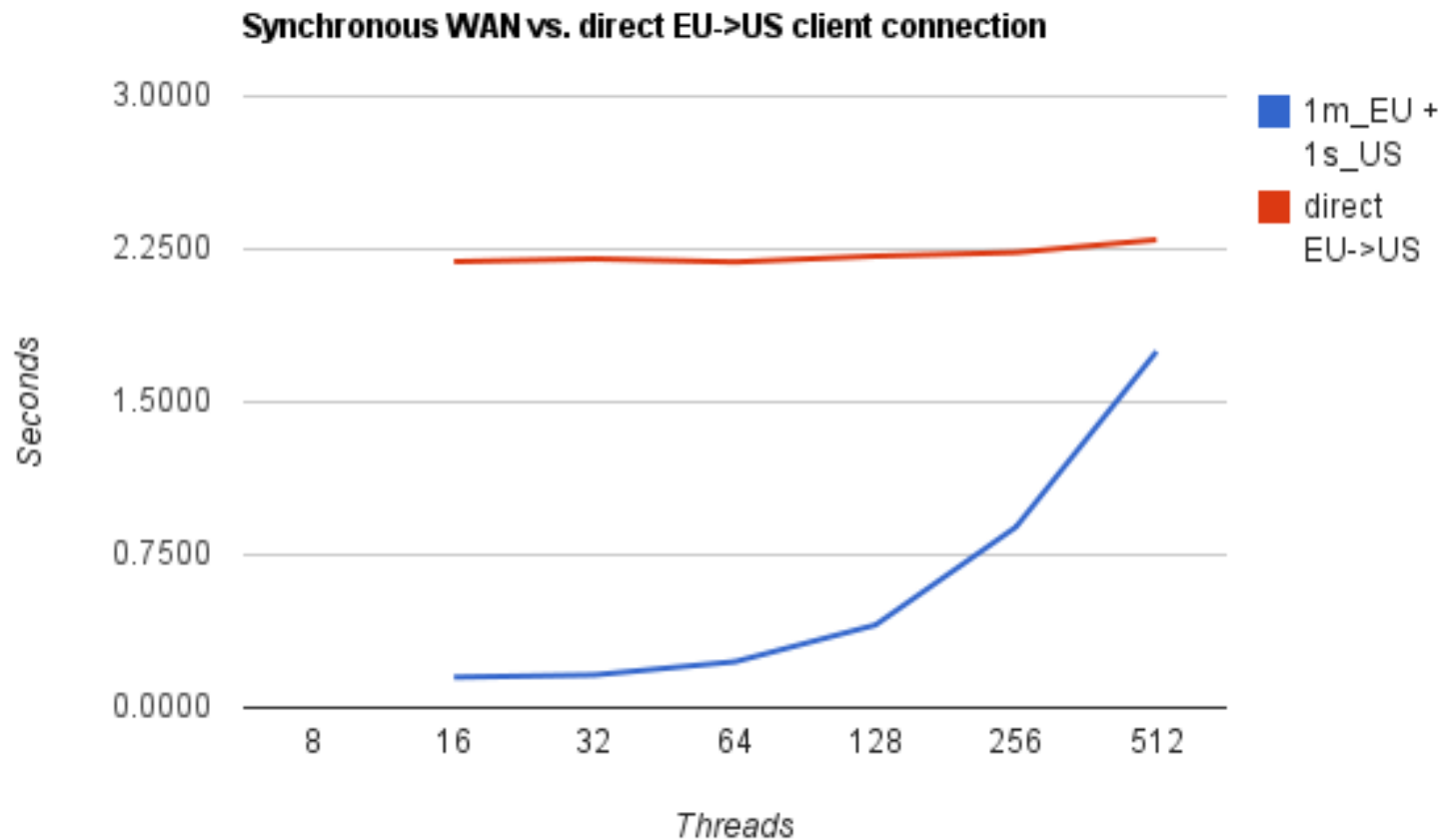


Synchronous WAN Replication



- Sysbench OLTP complex mode
- Amazon EC2

Synchronous WAN Replication



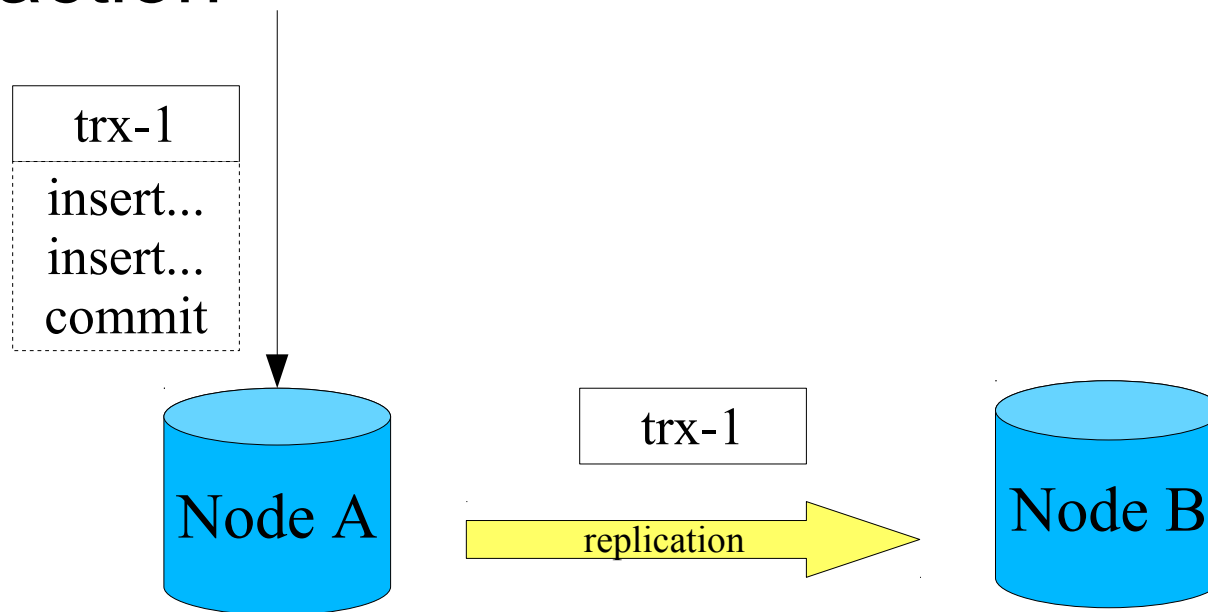
- Sysbench OLTP complex mode
- Amazon EC2

Why Multi-Master?

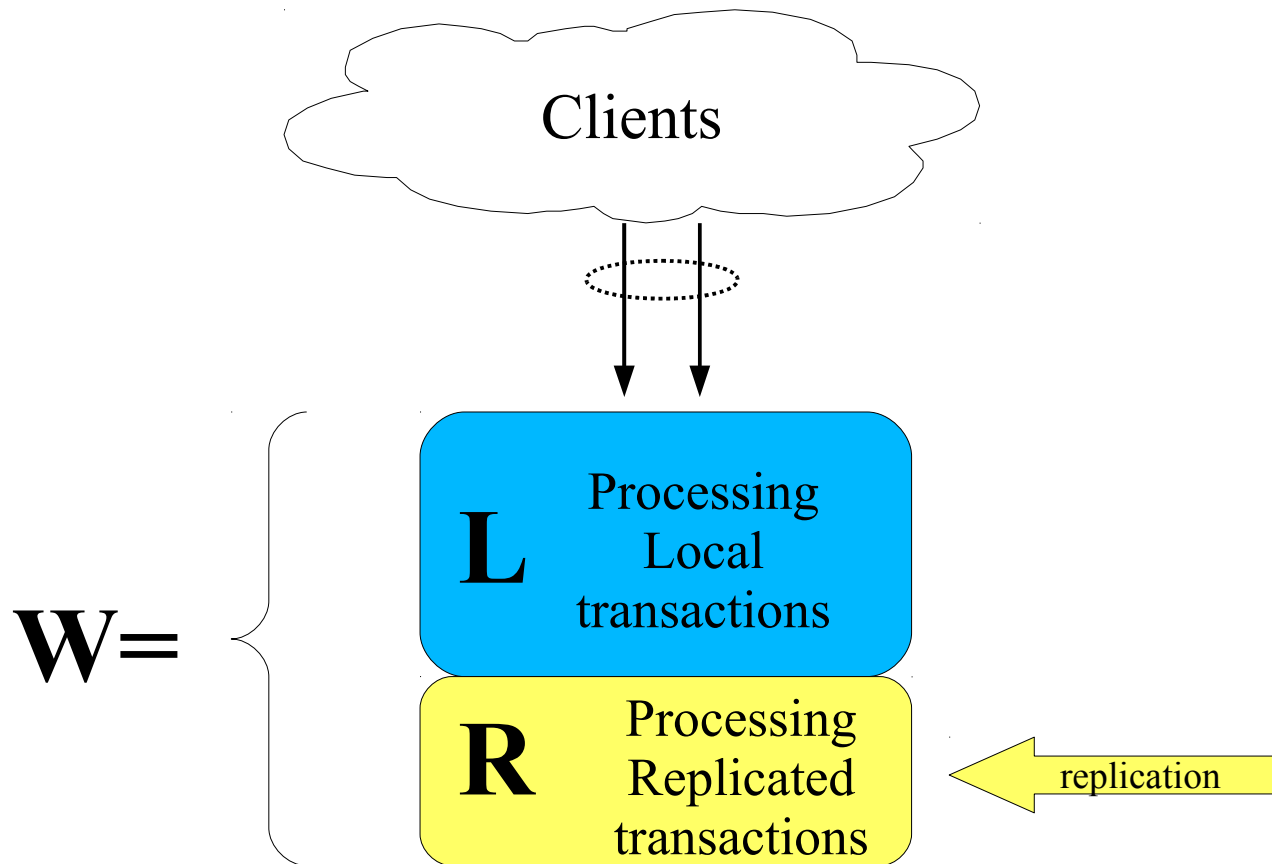
- Simple to use, less responsibilities in application side
- Local access in WAN cluster
- **Write Scalability**

Write Scalability

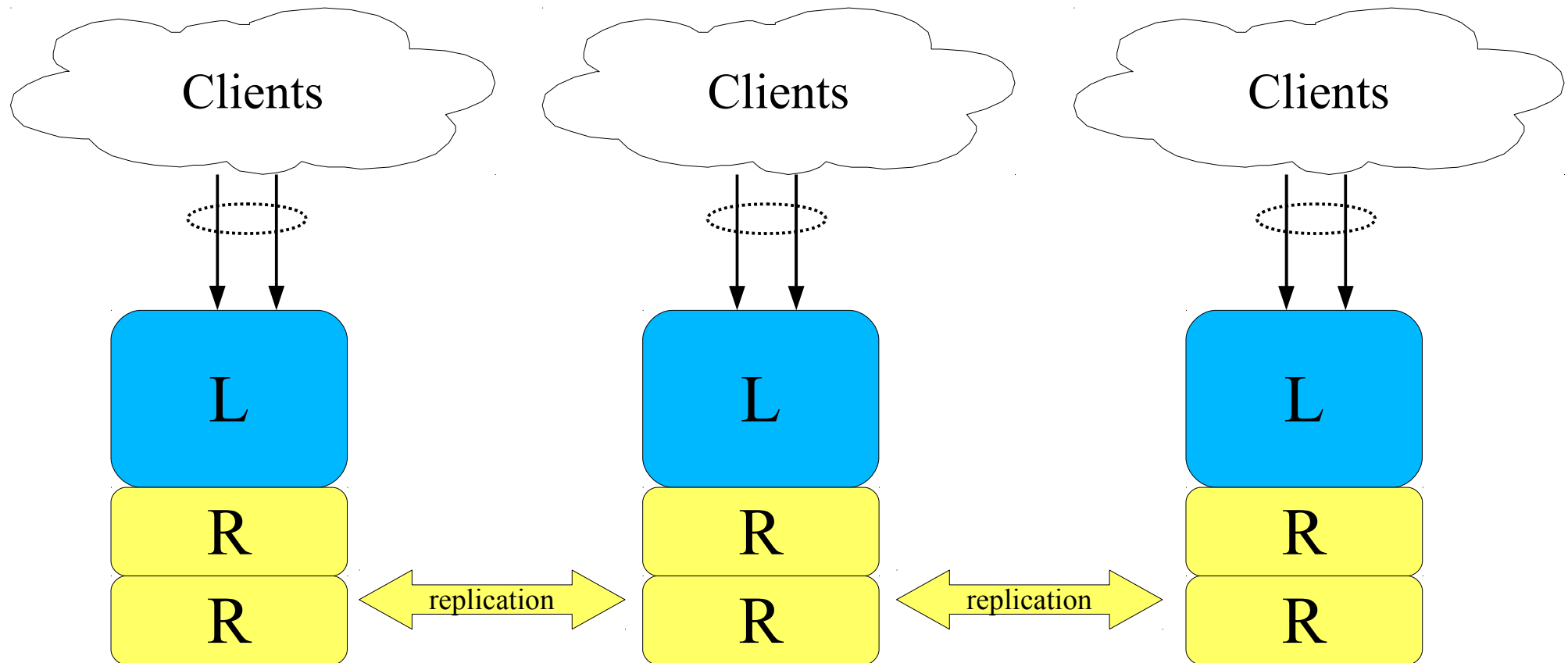
- Possible if applying of replicated transaction is faster than processing the original transaction



Write Scalability



Write Scalability



Scalability Arithmetics

W = work done in a node

L = work done processing local transactions

R = work done processing replicated transactions

$\alpha = R/L$ /* Replication coefficient */

$\varepsilon = L / W$ /* Node efficiency */

$S = n * \varepsilon$ /* Scalability */

$W = L + R * (n-1)$ /* Node performance */

$\Rightarrow W = L * (1 + \alpha (n-1))$

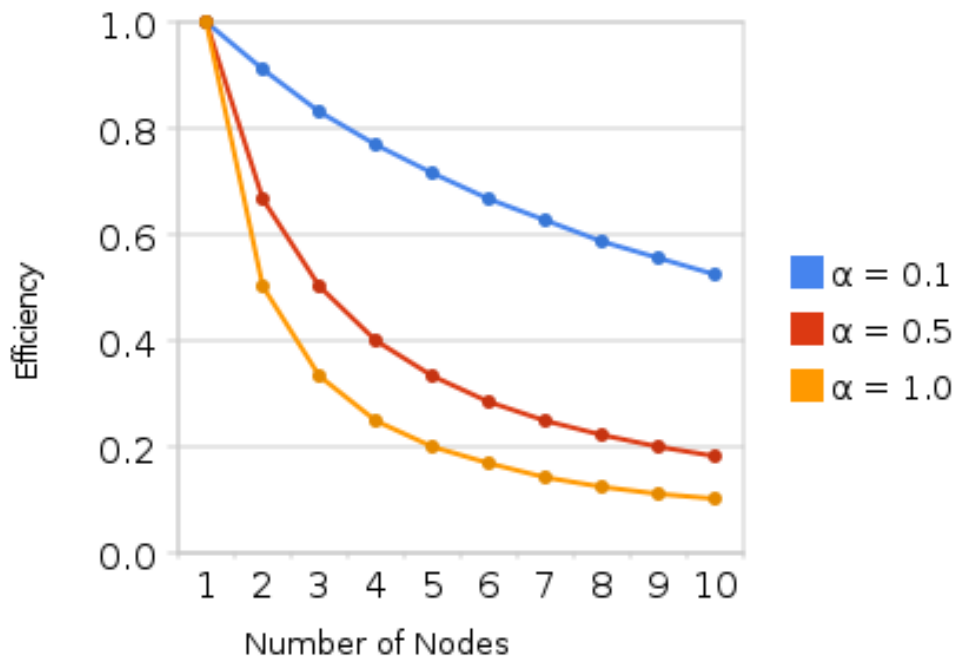
$\Rightarrow \varepsilon = 1 / (1 + \alpha * (n - 1))$

$\Rightarrow S = n / (1 + \alpha * (n - 1))$

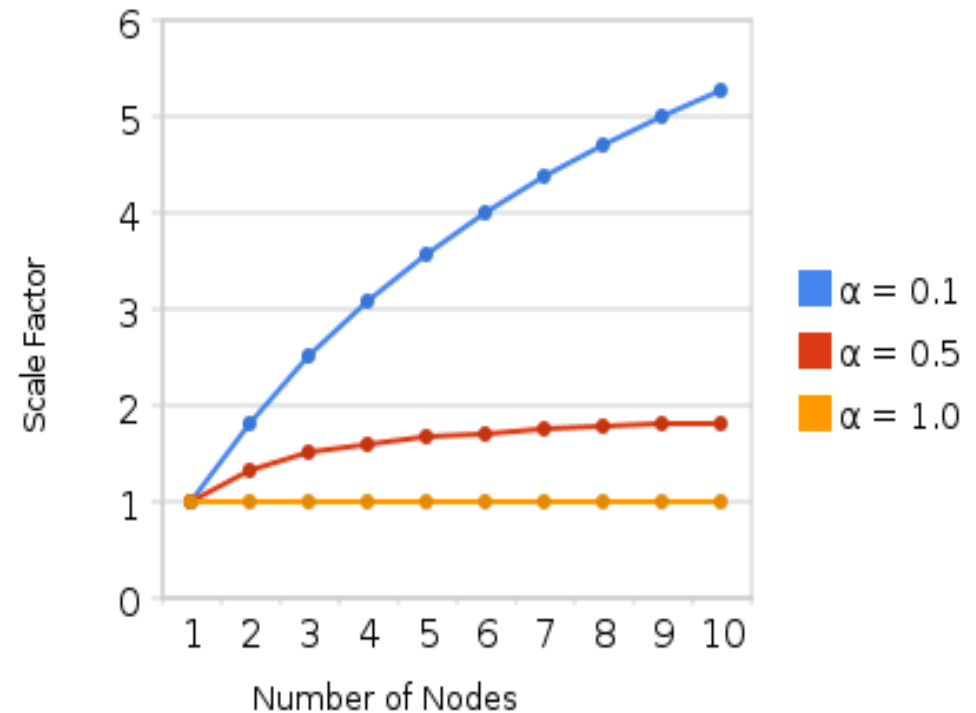
See details in Alexey's blog: <http://www.codership.com/content/multi-master-arithmetics>

Scalability Arithmetics

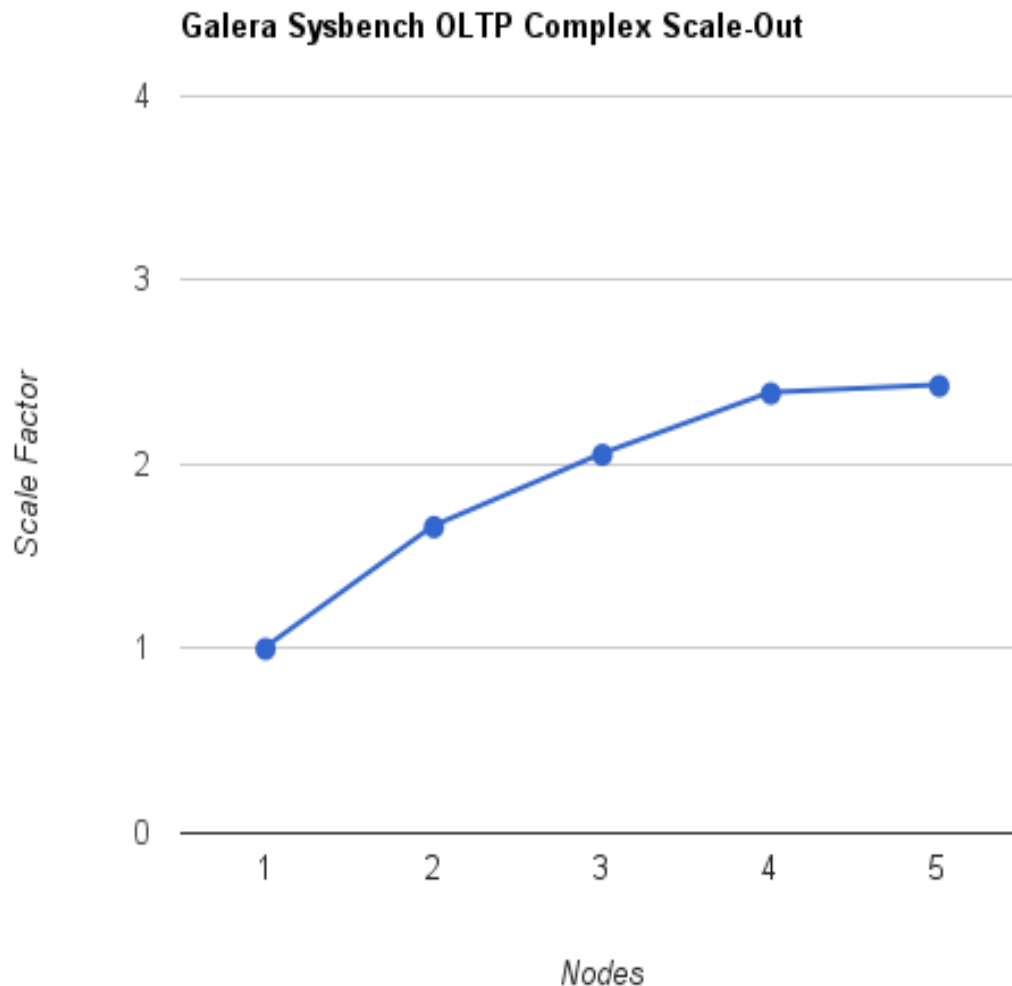
Node Efficiency in Multi-Master Cluster



Multi-Master Cluster Scalability



Sysbench OLTP Complex Scale-Out



- 24% write queries
- $\alpha \sim 9\%$
(estimated from CPU utilization)

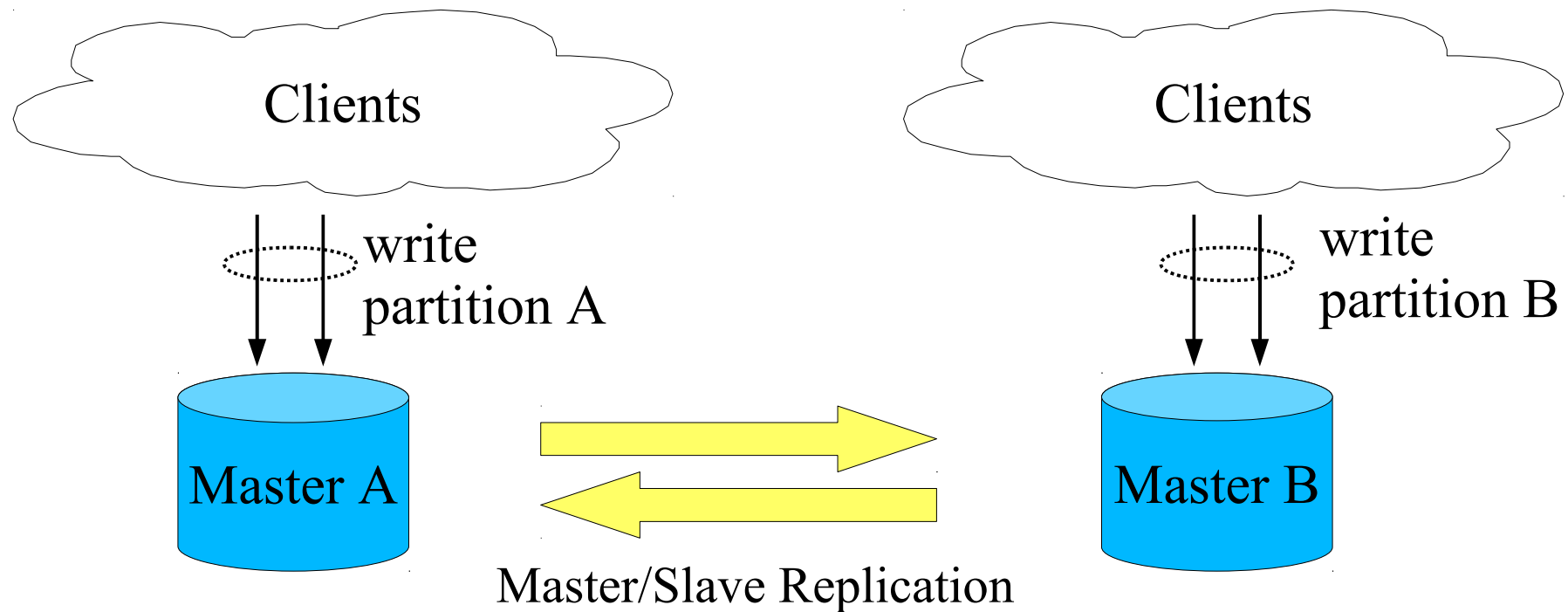
Multi-Master Approaches

- **Asynchronous multi-master**
 - Need to deal with the conflicts
 - × Preventing
 - × Detecting
 - × Resolving
- **Synchronous multi-master**
 - Concurrency control
 - × Pessimistic
 - × Optimistic

Asynchronous Replication

Asynchronous Multi-Master

store and forward



Asynchronous Multi-Master

- Topologies
 - Bi-Directional
 - Circular
 - Star topology
- Transaction Tracking
- Have to deal with conflicts
 - Detecting
 - Preventing
 - Resolving

Conflict Resolution

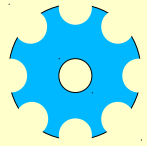
- Conflict types
 - Update conflict
 - Uniqueness conflict
 - Delete conflict
- Some conflicts can be resolved automatically
- ...but resolving is not always possible

=> application must be prepared to deal with async MM

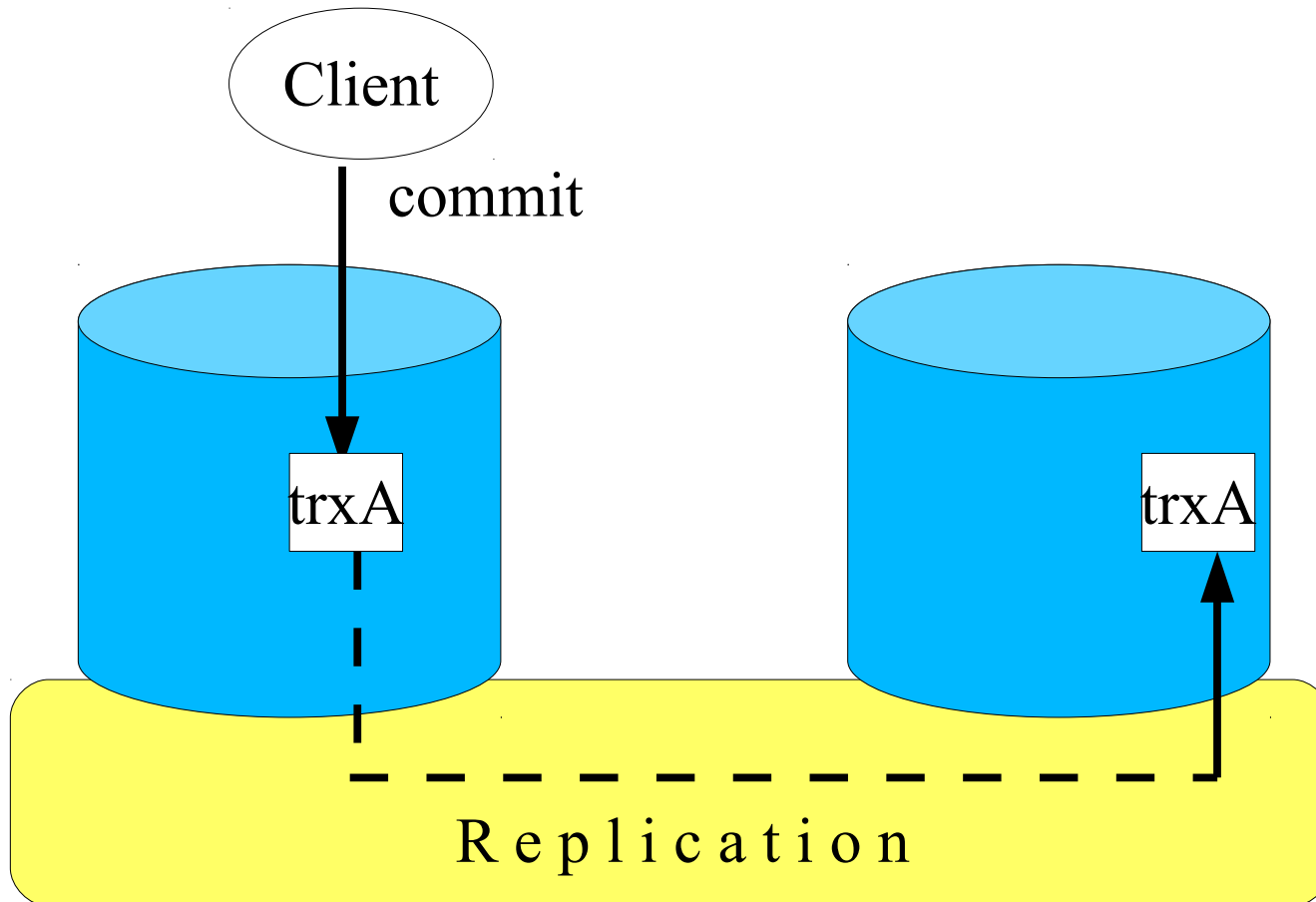
Conflict Preventing

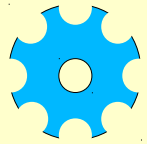
- Inserts are safe, if primary keys are guaranteed not to conflict
 - Use autoincrement options or sequences
 - Use node name as part of key
- Updating unique columns should be avoided
- Some times deleting can be avoided by just marking record as deleted
- Unsafe operations can be routed through one dedicated node (falling back to Master/Slave topology)

Synchronous Replication

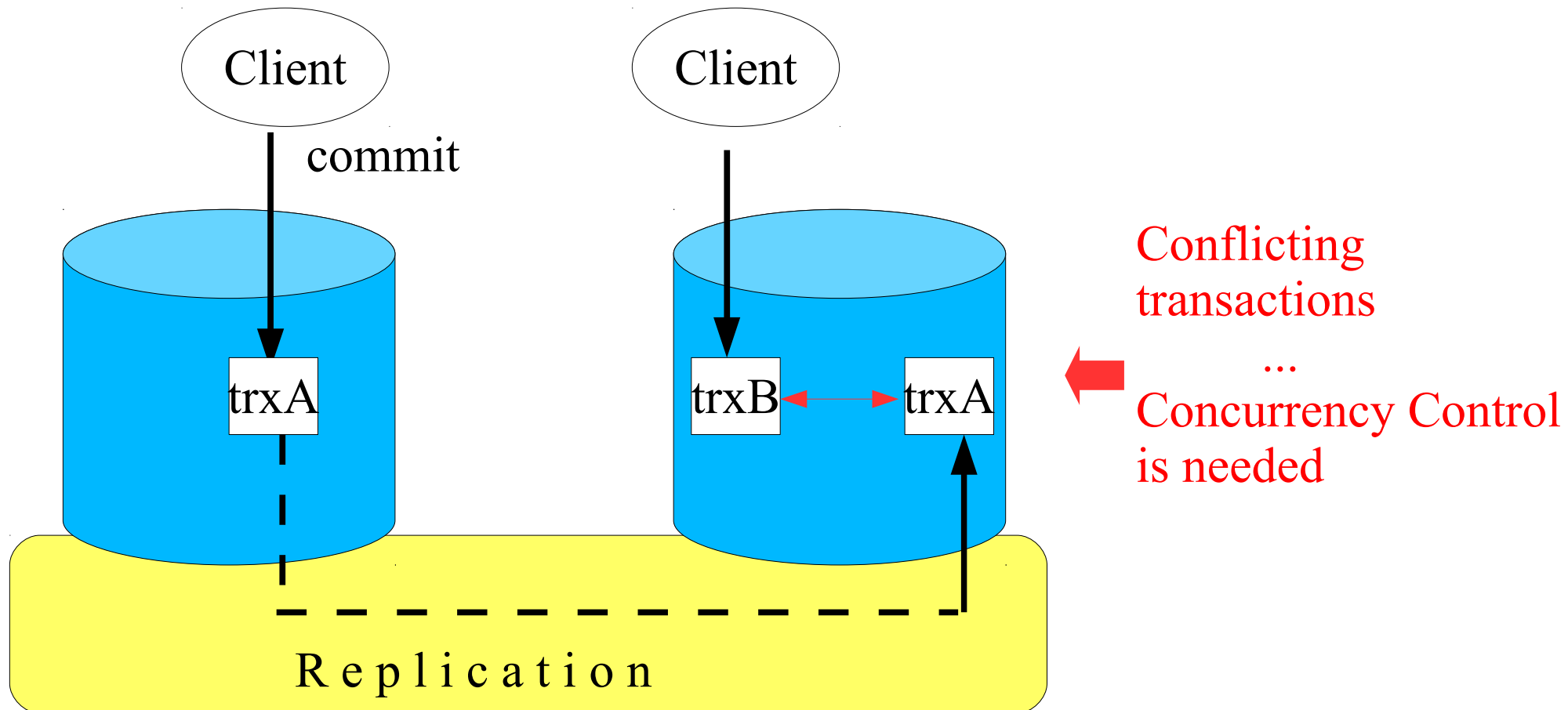


Synchronous Replication





Synchronous Replication

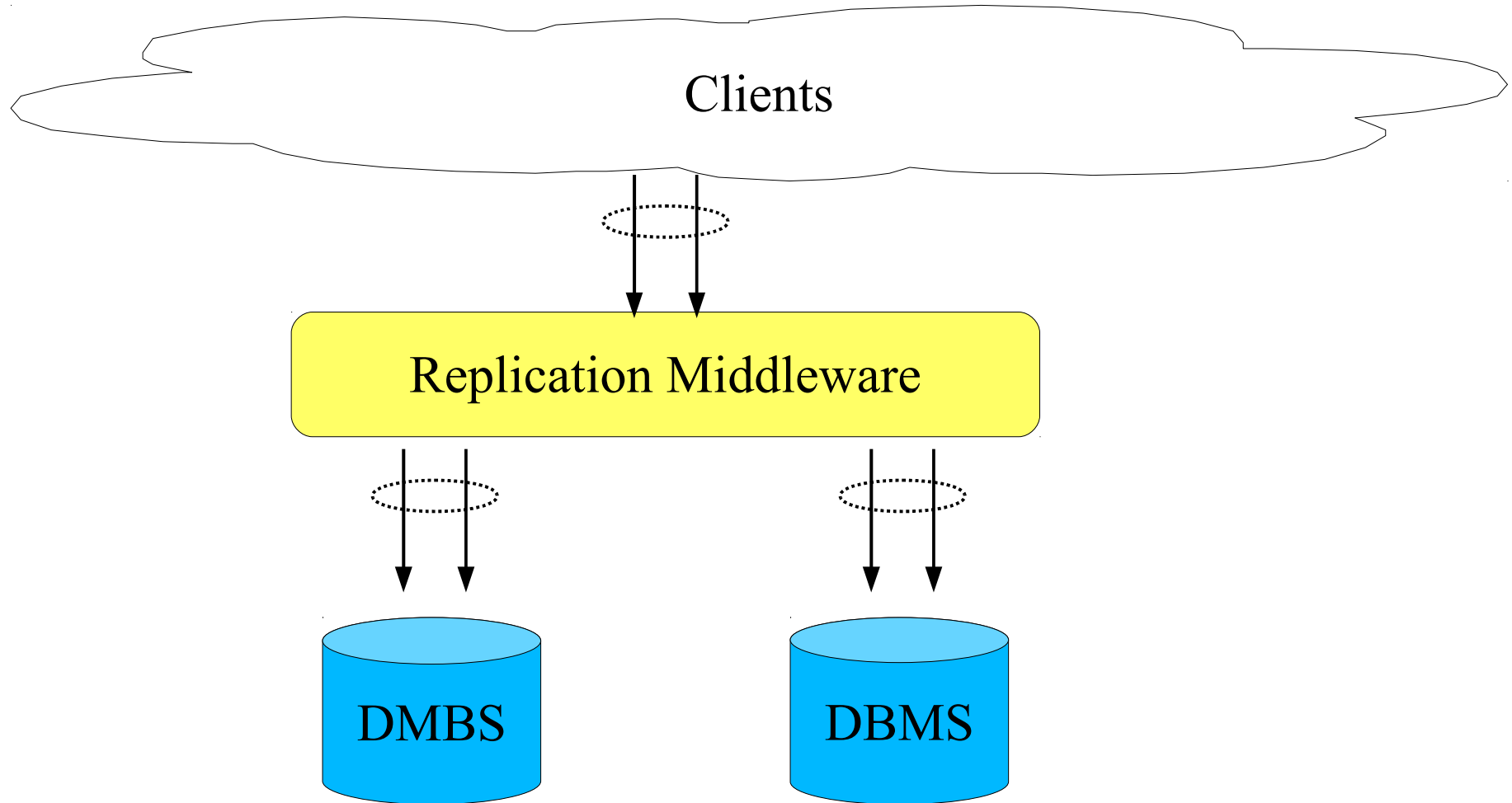


Concurrency Control

Optimistic vs Pessimistic

- Optimistic Concurrency Control
 - Transactions proceed independently in each cluster node assuming they can eventually commit
 - There can be cluster wide conflicts
 - Victim trx must abort
 - DEADLOCK error returned for cluster aborts
- Pessimistic Concurrency Control
 - Distributed locking, 2PC (or similar) to provide locking in all nodes

Middleware Replication



Multi-Master Approaches

Asynchronous

Conflict
Detecting

Conflict
Resolution

Synchronous

Optimistic

Pessimistic

Middleware

Locking

Multi-Master Solutions

MySQL Replication

- Supports ring topology
- Transaction cycles prevented
- MMM can be used when only one active master is needed

rubyrep

- Async replication with 2 masters
- PG and MySQL
- Custom conflict resolution

Drizzle Multi-Master

- Multiple masters replicating into one slave
- Centralized Backup Server

Tungsten

- Async replication
- Replication pipelines
- Rule based management
- Multi-Master within database/shard granularity
- Transaction tracking

Postgres-xc

- Sync multi-master
- Partitioning
- Benchmarking shows impressive scale factors
- Modules
 - Coordinator (intercepts calls and delegates to Data nodes)
 - Global Transaction Manager (grants global ID)
 - Data node (SQL processing)

MySQL Cluster

- NDB Engine
- Synchronous Multi-Master with 2PC
- Inbuilt partitioning
- Supports also async replication between MySQL Clusters with conflict resolution

Galera

- Synchronous,
- optimistic CC, certification based
- Generic replication
 - Replication API
 - MySQL/InnoDB, MariaDB supported
 - PG version under development

Postgres-R

- Synchronous multi-master replication
- Optimistic CC, certification based
- Same principles as in Galera replication

Multi-Master Approaches

Asynchronous

Synchronous

Conflict
Detecting

Conflict
Resolution

Optimistic

Pessimistic

MySQL Replication

Tungsten

Bucardo

Drizzle MM

RAC

MySQL/Cluster

Active Directory

rubyrep

Postgres-R

Galera

Middleware

Locking

Sequoia
m/cluster
HA/JDBC

MySQL Cluster
RAC
Postgres-xc

Summary

- Multi-Master cluster can solve many issues with Master/Slave replication
- Write scalability has hard limits => partitioning is needed to gain “ultimate speed”
- Asynchronous MM requires tuning in the application
- There are several projects providing solutions to Multi-Master replication

Galera Project

- Open source project
- Synchronous Multi-Master with Optimistic CC
- Public MySQL/Galera releases since Jan 2009
- MySQL/Galera 0.8 shipping out
- MariaDB version by MontyProgram
- PostgreSQL project started

Get in Touch!

codership

- R&D consulting services
- Galera Support
- Downloads available: <http://www.codership.com>
- info@codership.com
- Mailing list: codership-team@googlegroups.com