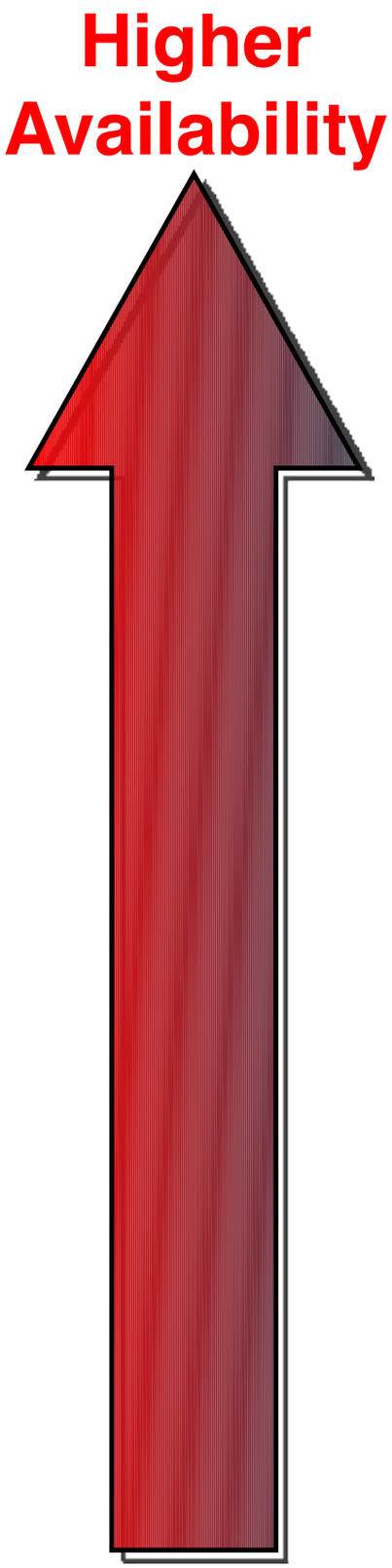
Architectures Haute-Dispo

Joffrey MICHAÏE
Consultant MySQL

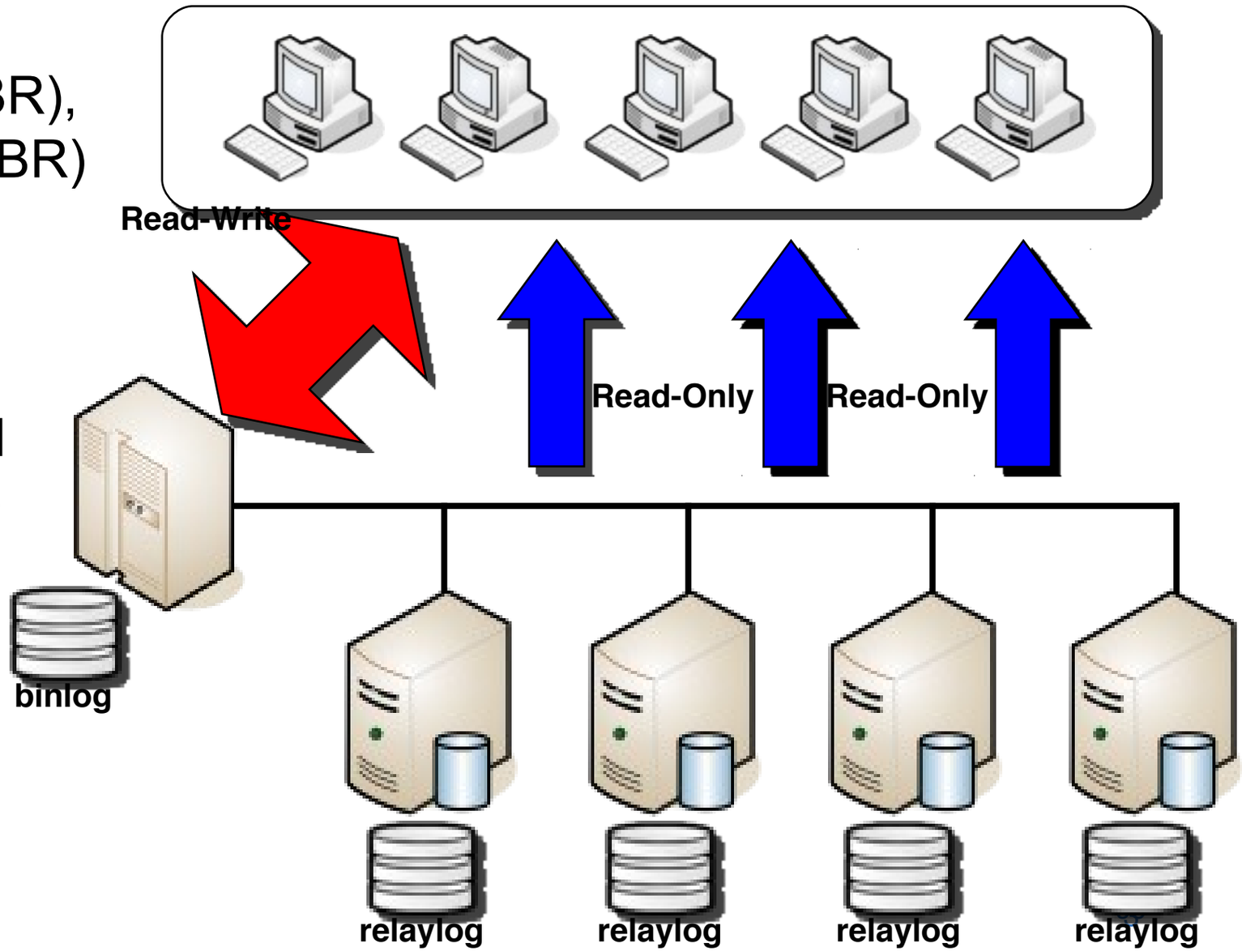
High Availability with MySQL



- Shared nothing distributed cluster with MySQL Cluster
- Storage snapshots for disaster recoveryGeographical Replication for disaster recovery
- Virtualised Environments
- Active/Passive Clusters + Local Asynchronous ReplicationActive/Passive Clusters through shared storage
- DRBD + Local Asynchronous Replication
- Synchronous Replication through DRBDLocal Semi-synchronous ReplicationLocal Asynchronous Replication

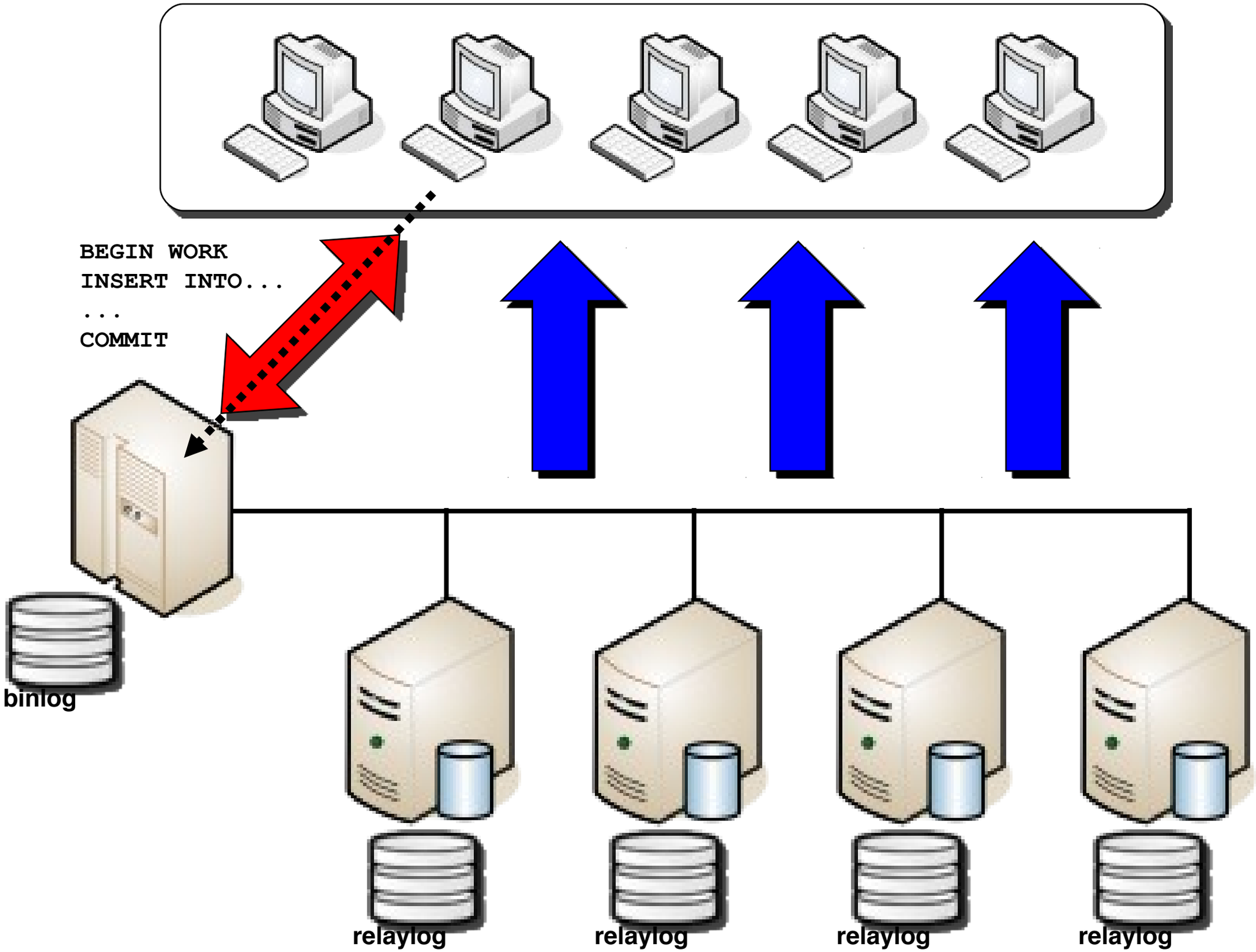
Local Asynchronous Replication

- Data written on the master is written into the binary log
- The I/O thread on the slave collects logs from the master binary log and writes a relay log on the slave
- The SQL thread on the slave reads the relay log and apply the writes on the slave
- Slave writes are optionally added to the binary log on the slave
- It can be statement-based (SBR), row-based (RBR) or mixed (MBR)
- In case of fault:
  - The master server is taken down
  - The slave server is updated up to the last position in the relay log
  - The clients point at the designated slave server
  - The designated slave server becomes the master server



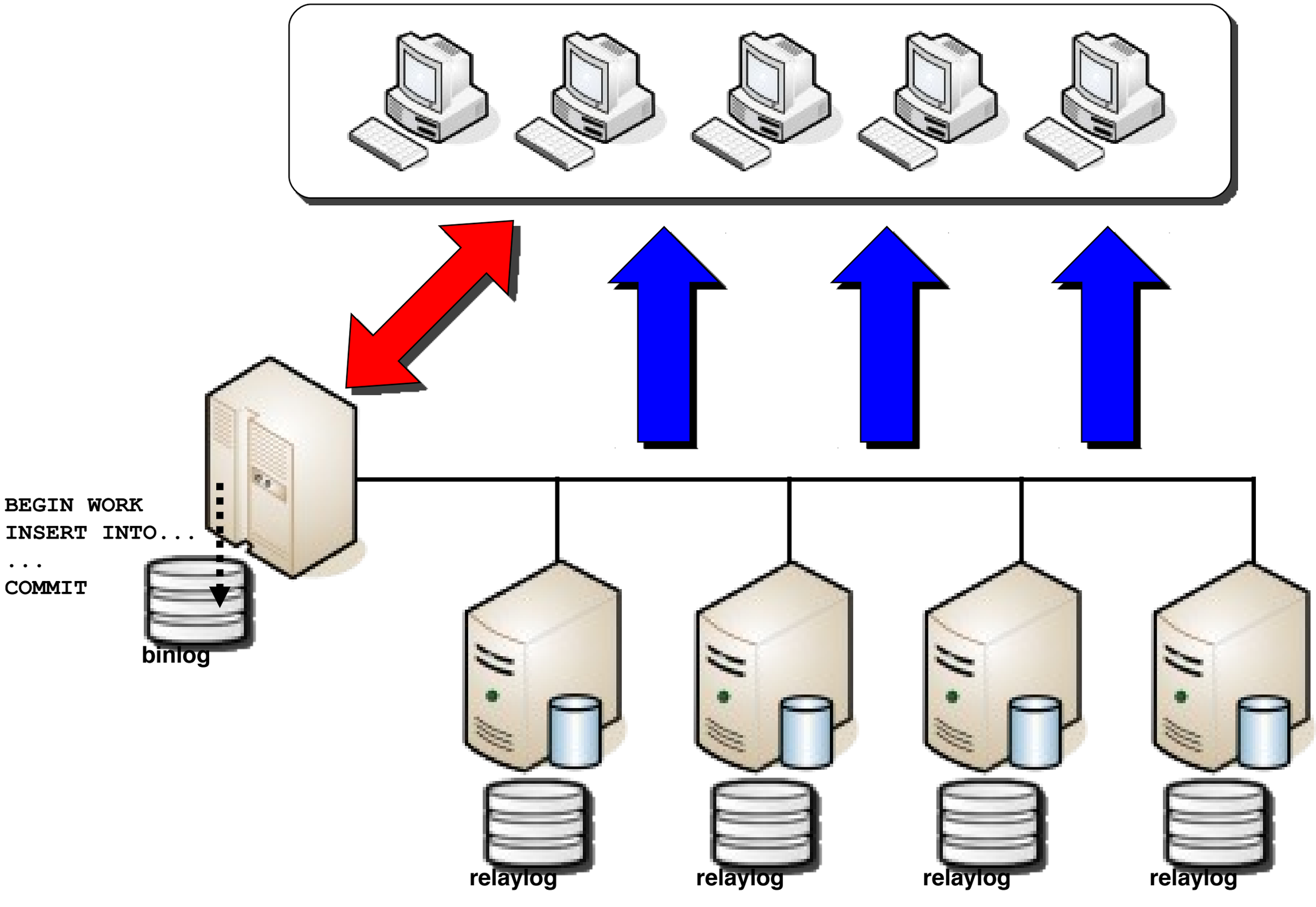
Local Asynchronous Replication

1. A write transaction is sent to the Master



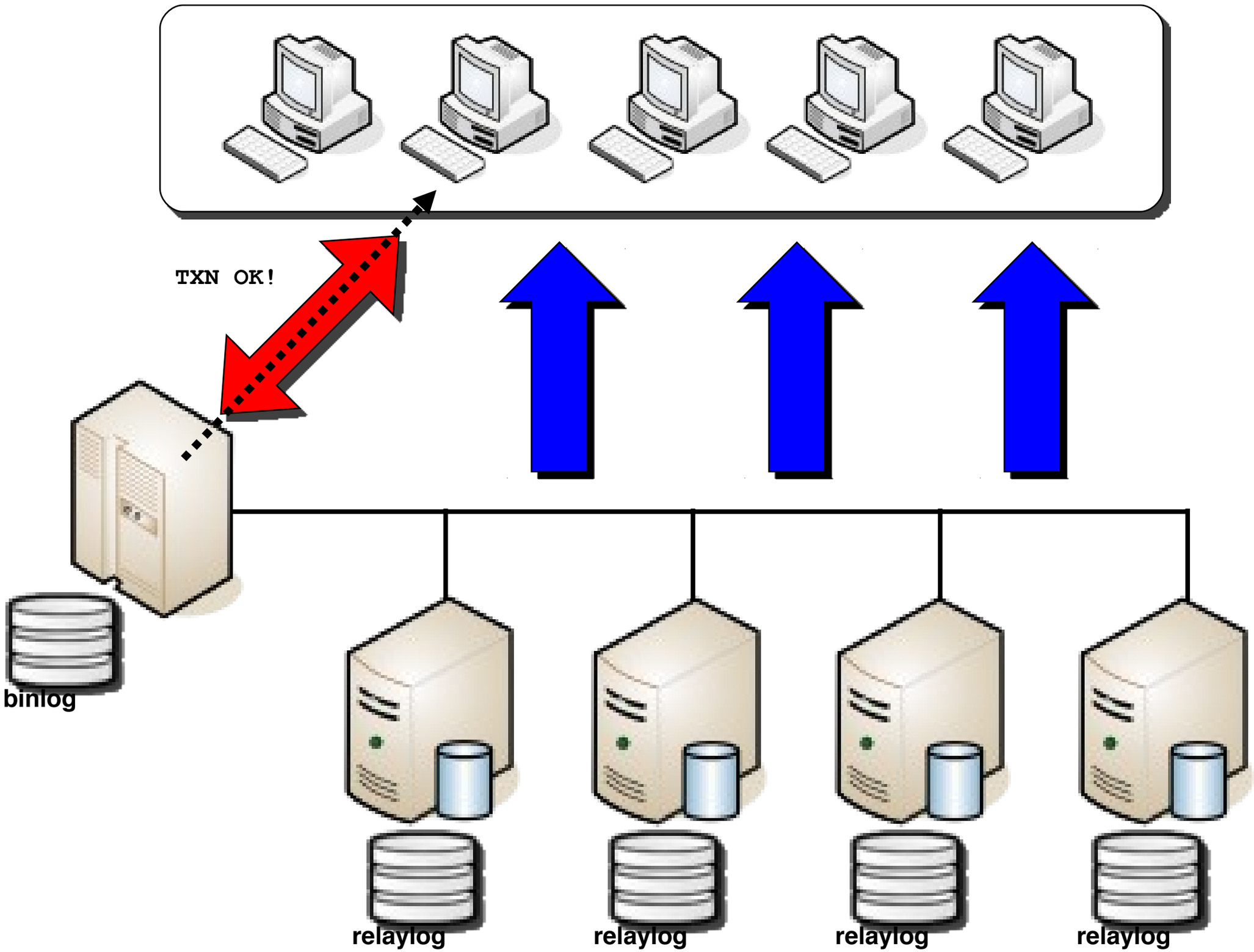
Local Asynchronous Replication

2. The transaction is stored into the binlog



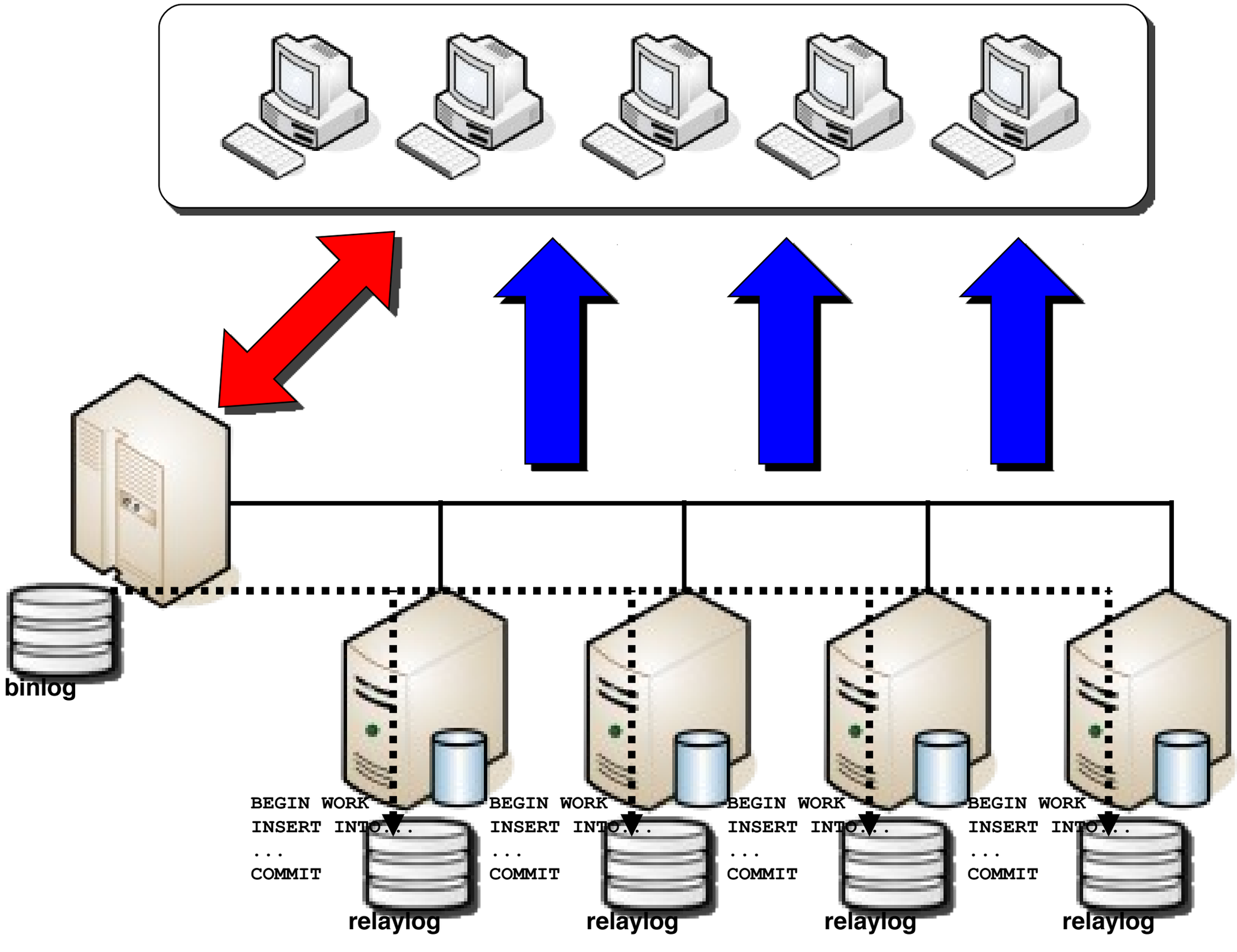
Local Asynchronous Replication

3. The client receives the ACK from the server



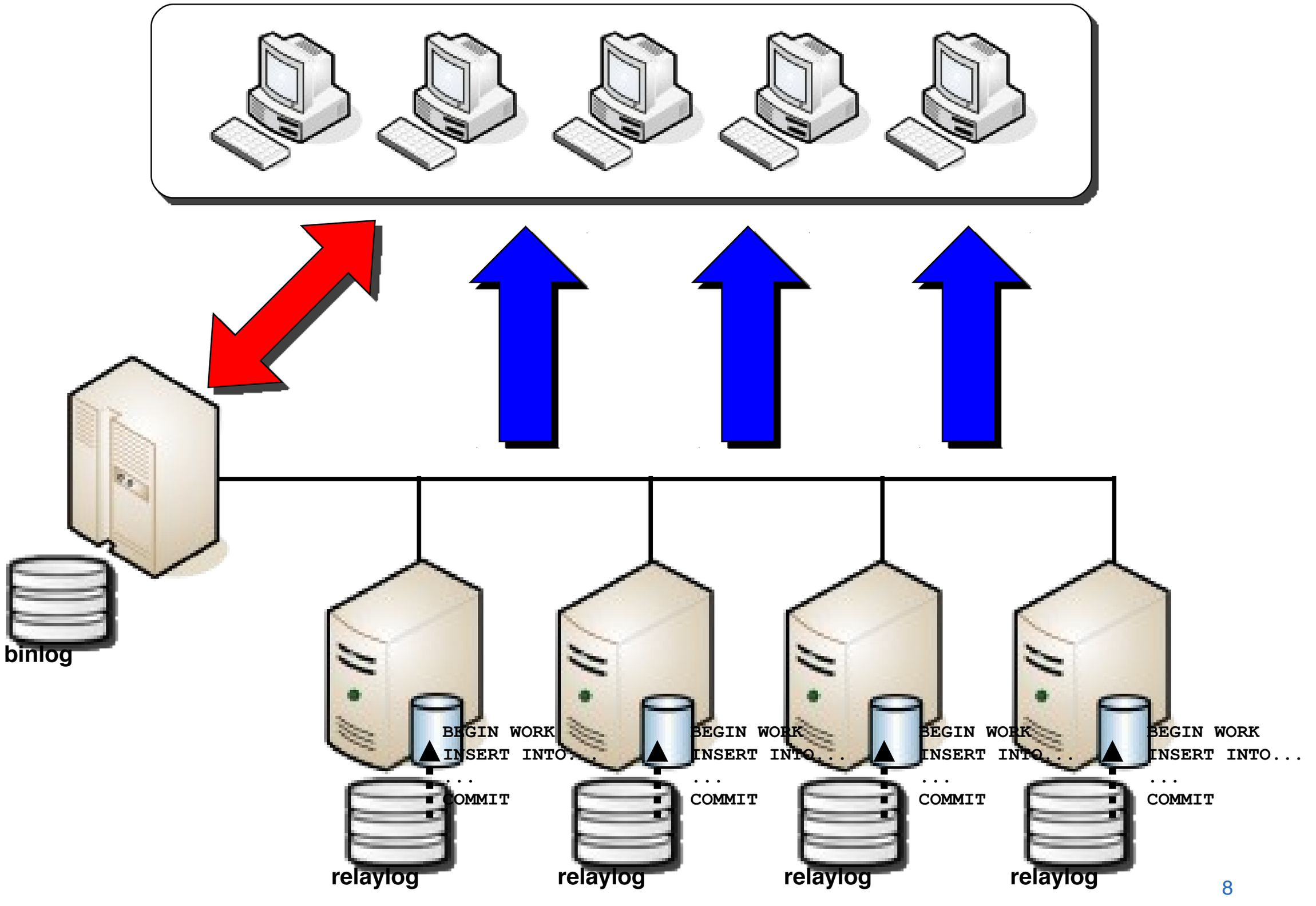
Local Asynchronous Replication

4. The I/O thread on the slave(s) pulls the transaction and it stores it in the relaylog



Local Asynchronous Replication

5. The SQL thread reads the transaction from the relay log and applies the transaction on the slave(s)



Local Asynchronous Replication

Typical Usage

- High availability with manual failover
  - Easy to implement and maintain, provides a good level of availability
- Non-intrusive backup
  - Backups can be executed on a slave server
  - The binlog and provides incremental backup and possible point-in-time recovery
- Scalability for read-intensive applications
  - Read-only transactions can be executed on the slave servers
- Systems and applications upgrades
  - Chains of Master/Slaves reduce to virtually 0 the downtime for upgrades

Local Asynchronous Replication

Other Features

- Out-of-the box with MySQL Server
- Platform independent
  - Master and Slaves can be heterogeneous in OS and version
- Storage engine independent
  - Master and Slaves can be heterogeneous in storage engines
- Multiple topologies
- Works with transactional and non-transactional engines
- Advanced settings can define a selection of data to replicate

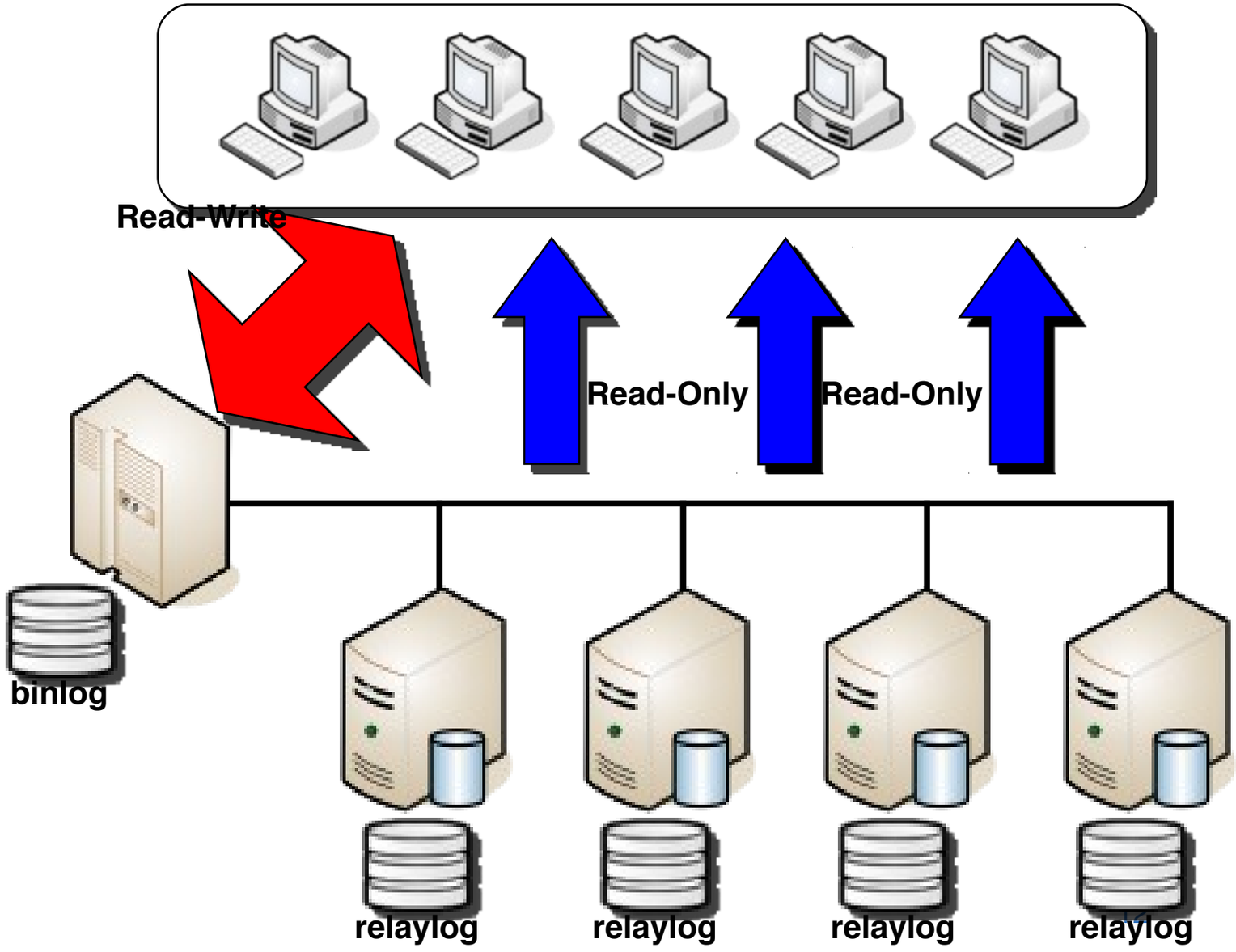
Local Asynchronous Replication

Other Features

- Out-of-the box with MySQL Server
- Platform independent
  - Master and Slaves can be heterogeneous in OS and version
- Storage engine independent
  - Master and Slaves can be heterogeneous in storage engines
- Multiple topologies
- Works with transactional and non-transactional engines
- Advanced settings can define a selection of data to replicate

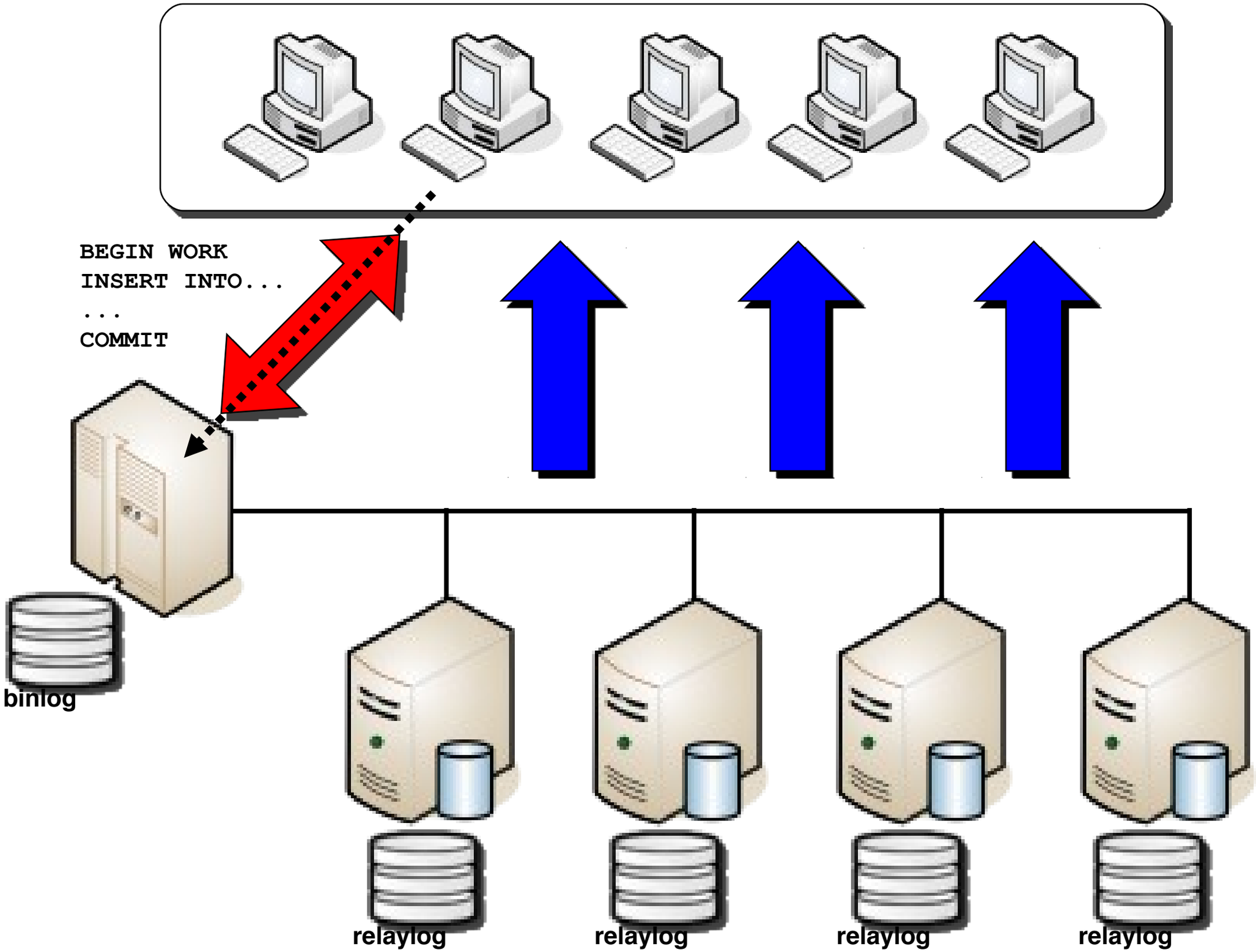
Local Semi-Synchronous Replication

- One or more slaves are defined as “semi-synchronous servers”
- the Master server waits until the I/O thread on one of the semi-sync slave(s) has flushed the transaction to disk, or until it receives a timeout, then it returns the ACK to the application.
- In case of fault:
  - The master server is taken down
  - The slave server is updated up to the last position in the relaylog (the update will be fast)
  - The clients point at the designated slave server
  - The designated slave server becomes the master server



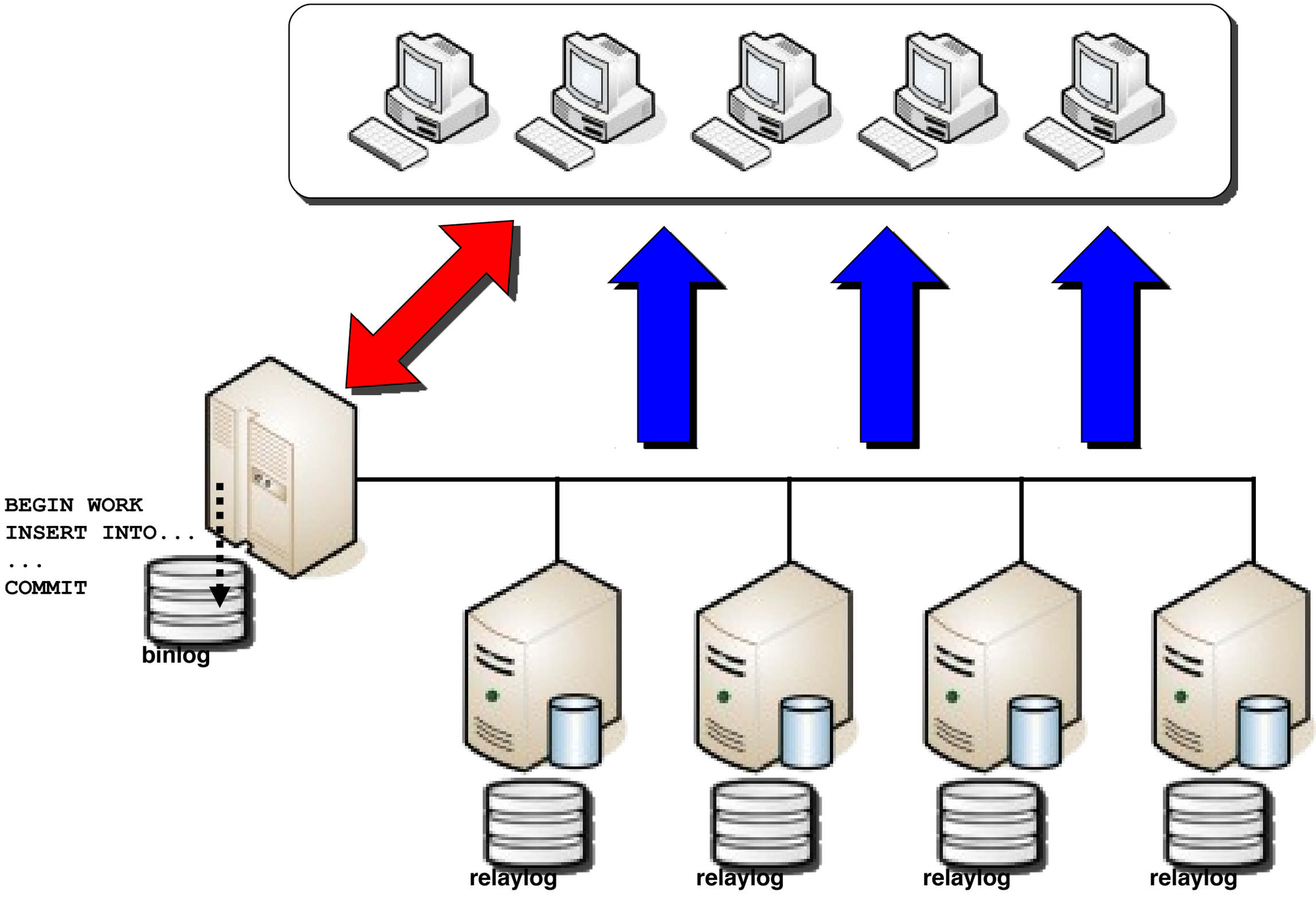
Local Semi-Synchronous Replication

1. A write transaction is sent to the Master



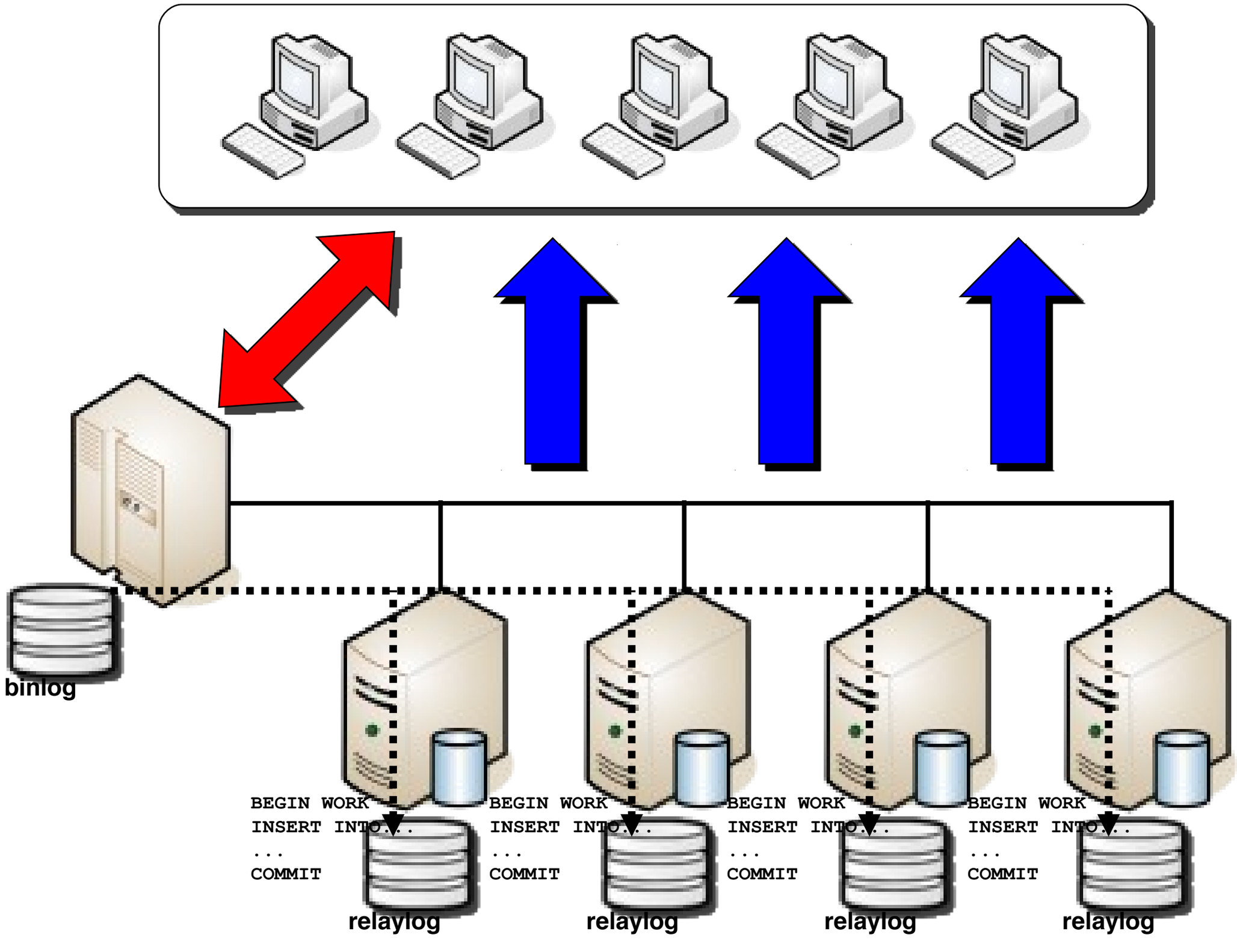
Local Asynchronous Replication

2. The transaction is stored into the binlog



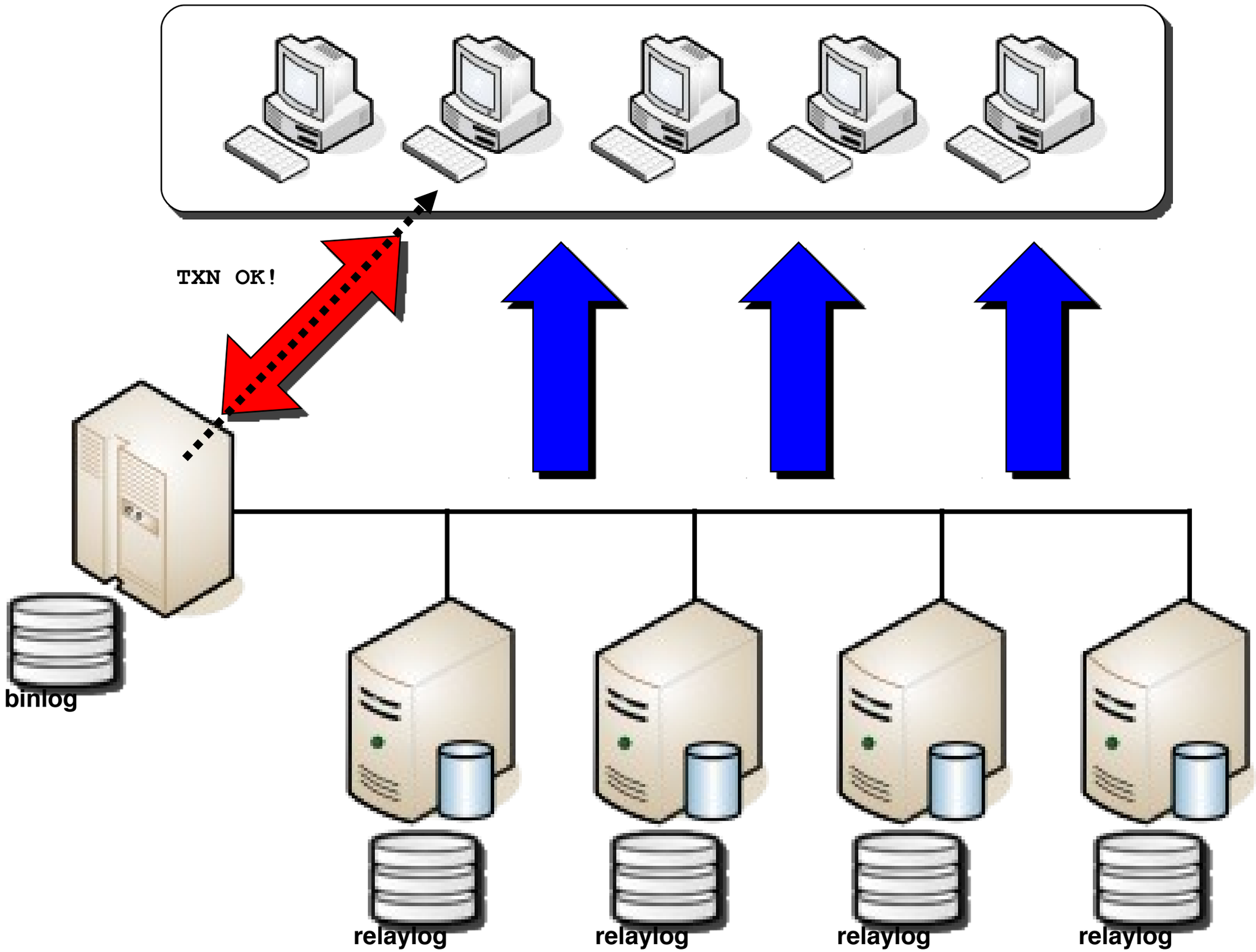
Local Asynchronous Replication

3. The I/O thread on the slave(s) pulls the transaction and it stores it in the relaylog



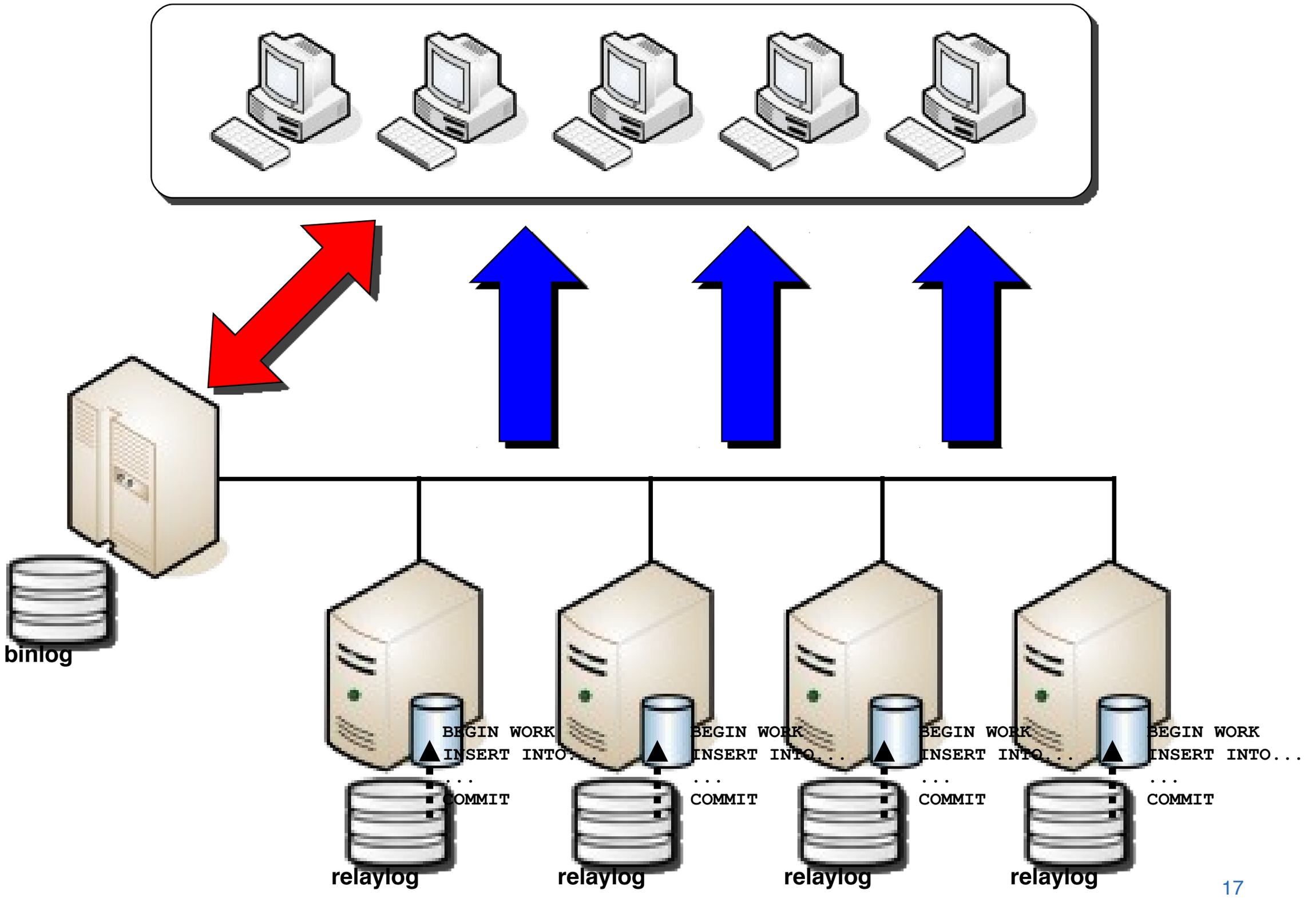
Local Asynchronous Replication

4. The client receives the ACK from the server



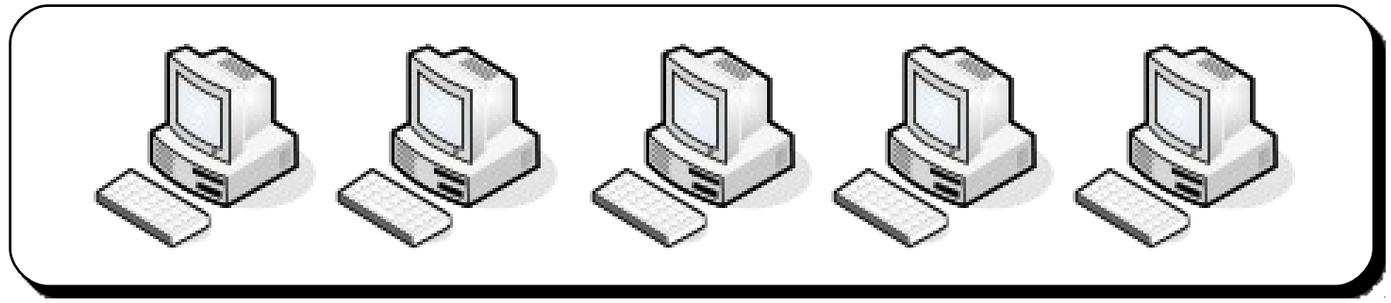
Local Asynchronous Replication

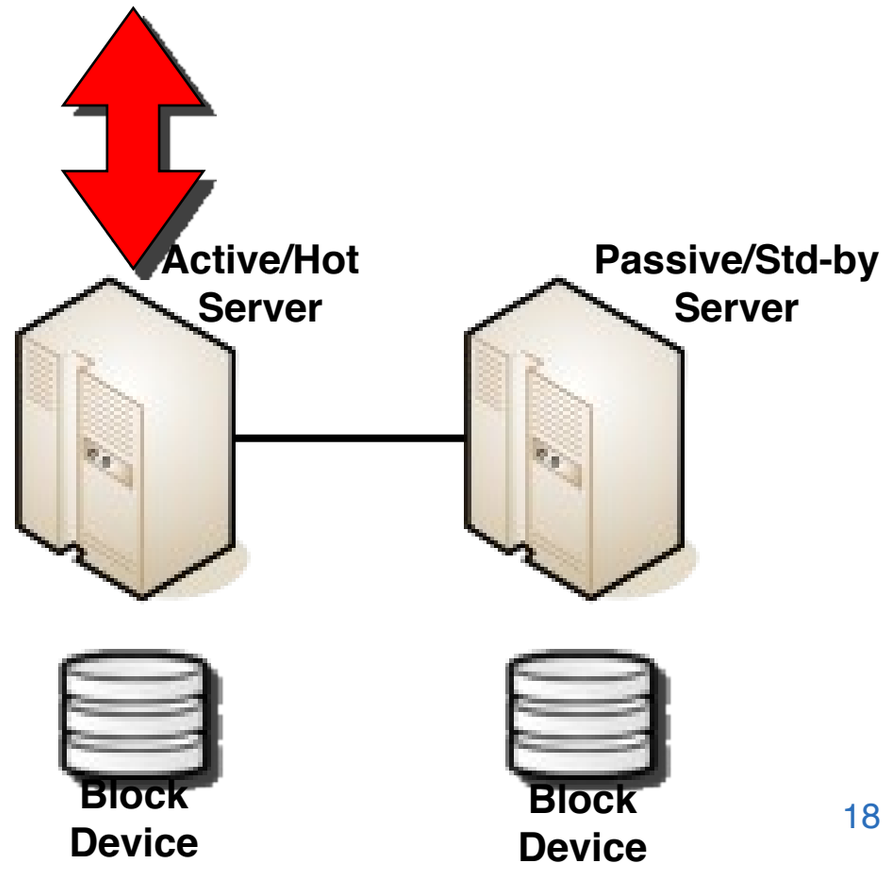
5. The SQL thread reads the transaction from the relay log and applies the transaction on the slave(s)



Synchronous Replication through DRBD

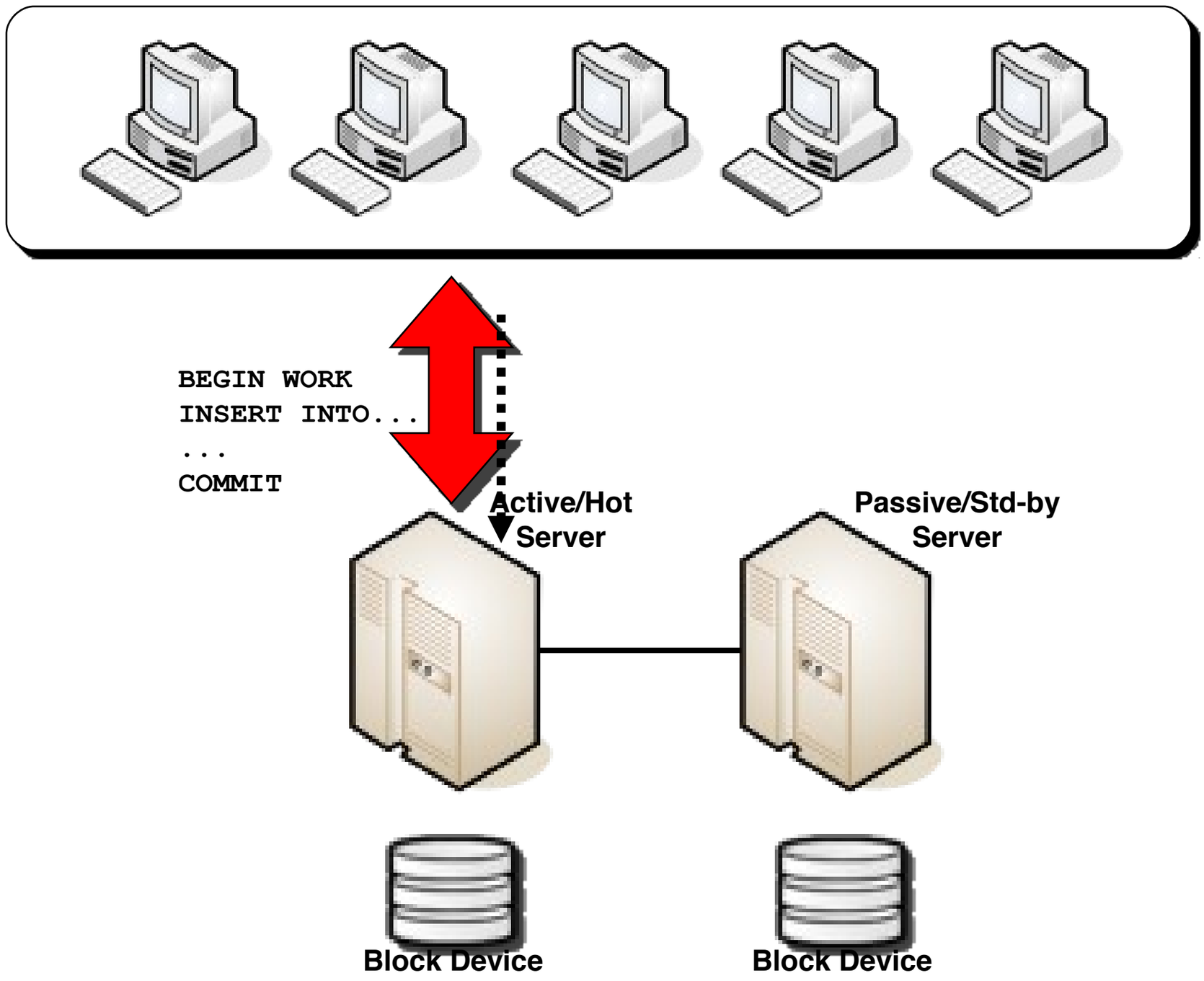
- One server is designated as “active” and another server as “passive” or “stand-by”
- The active server “owns” a virtual IP address. Applications refer to the virtual IP address to connect to the database server.
- Data is synchronously replicated at block level from the active to the stand-by server
- In case of fault:
  - The master server is taken down
  - The block device on the stand-by server is mounted
  - The mysqld on the stand-by server starts
  - The virtual IP address moves to the stand-by server
  - The stand-by server becomes active





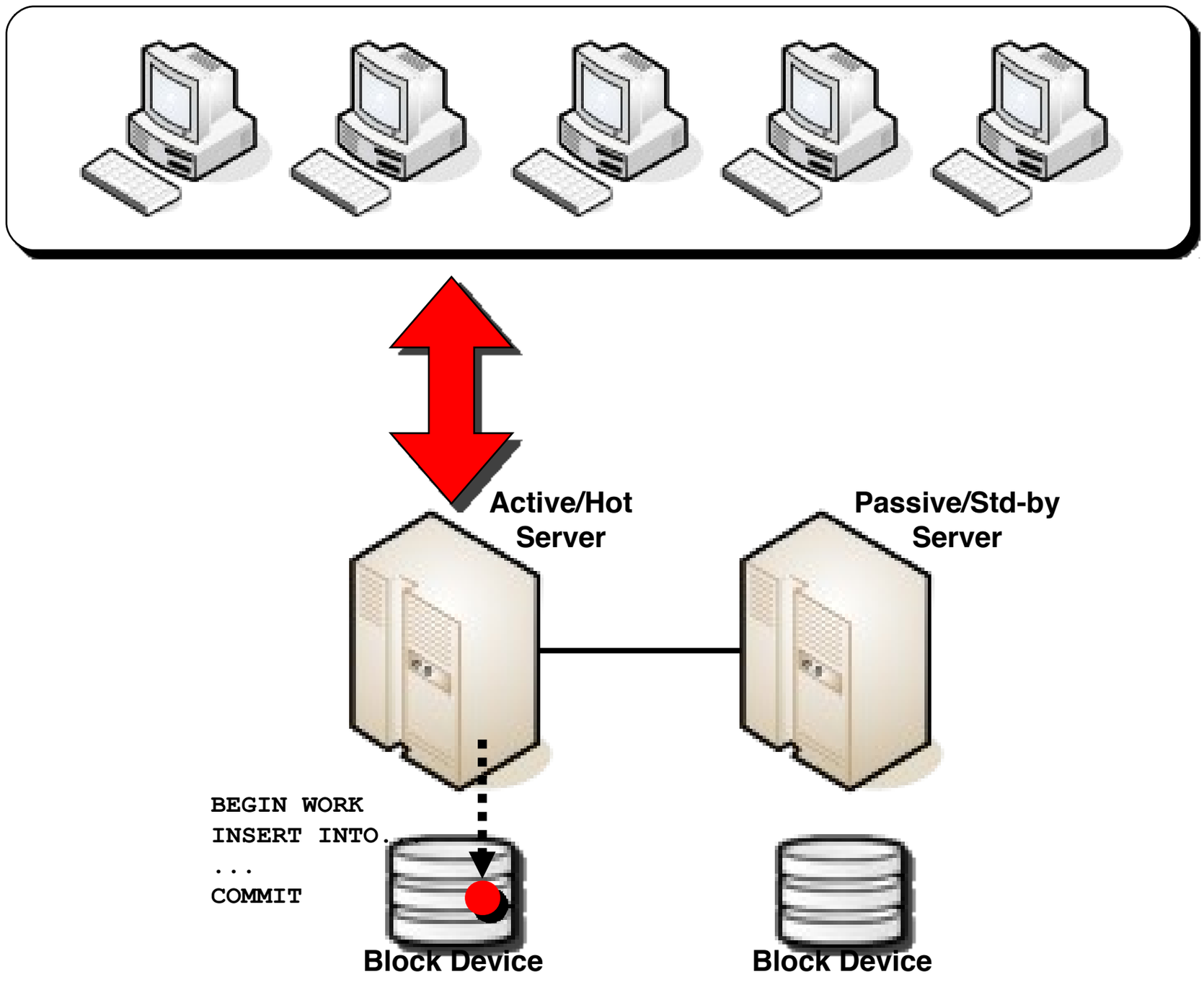
Synchronous Replication through DRBD

1. A write transaction is sent to the active Server



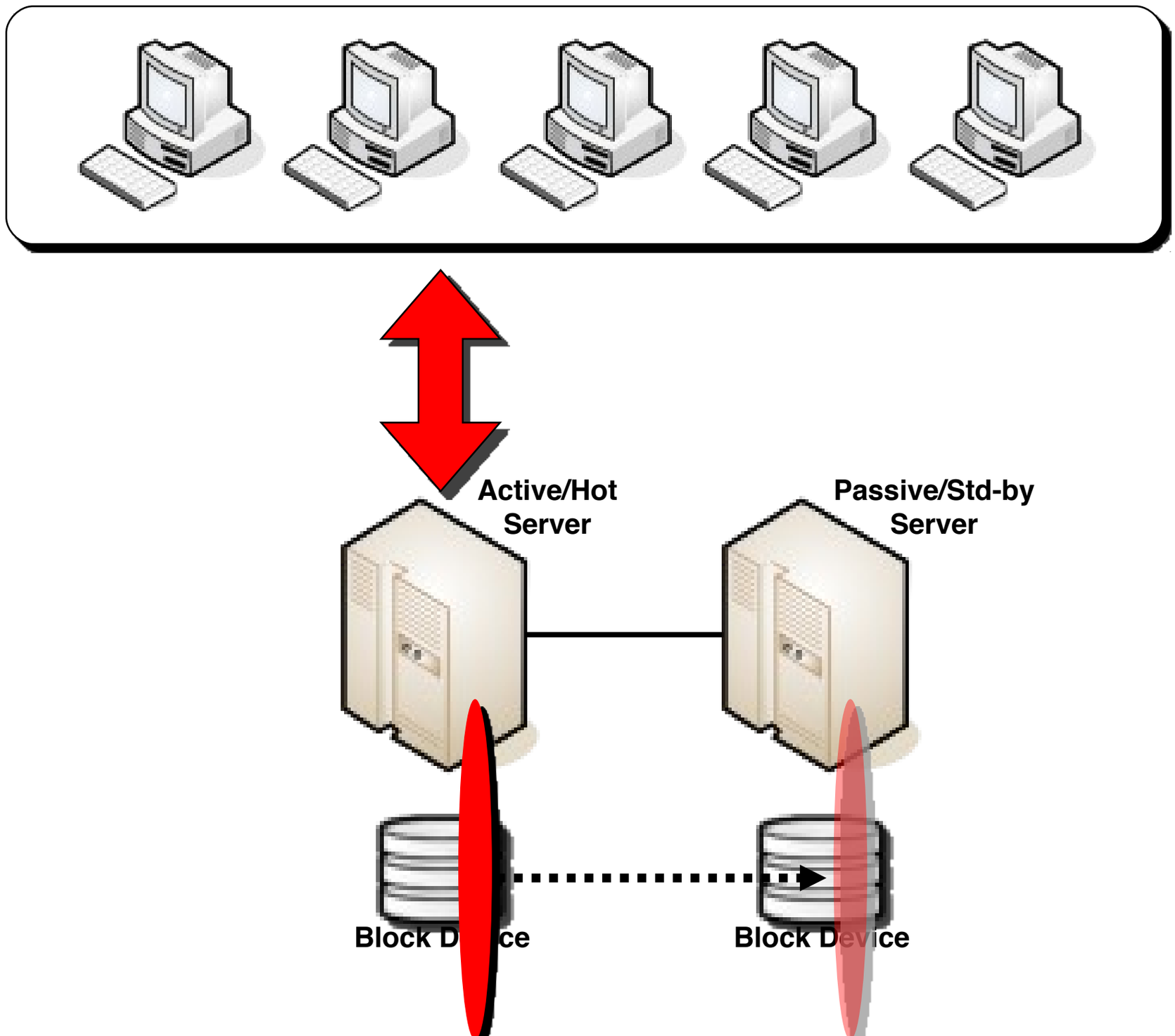
Synchronous Replication through DRBD

2. The transaction is written to the block device



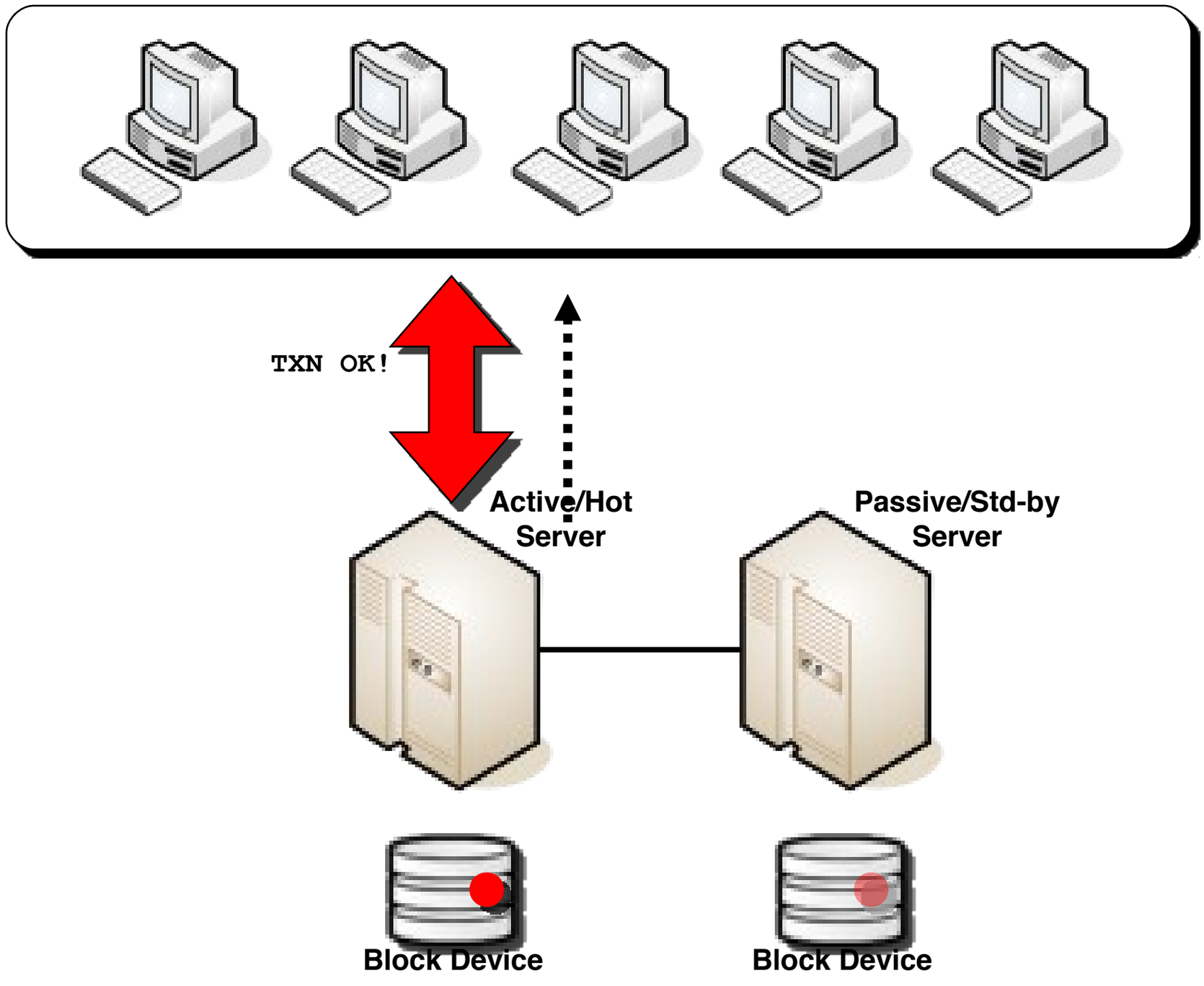
Synchronous Replication through DRBD

3. The blocks modified by the transaction are replicated to the stand-by server



Synchronous Replication through DRBD

4. The client receives the ACK from the server



Synchronous Replication through DRBD

Typical Usage

- High availability with automatic failover
  - Fully redundant solution with no loss of data and no single points of failure
- Point-in-time snapshots
  - One or more servers can be updated at regular intervals to provide historical analysis

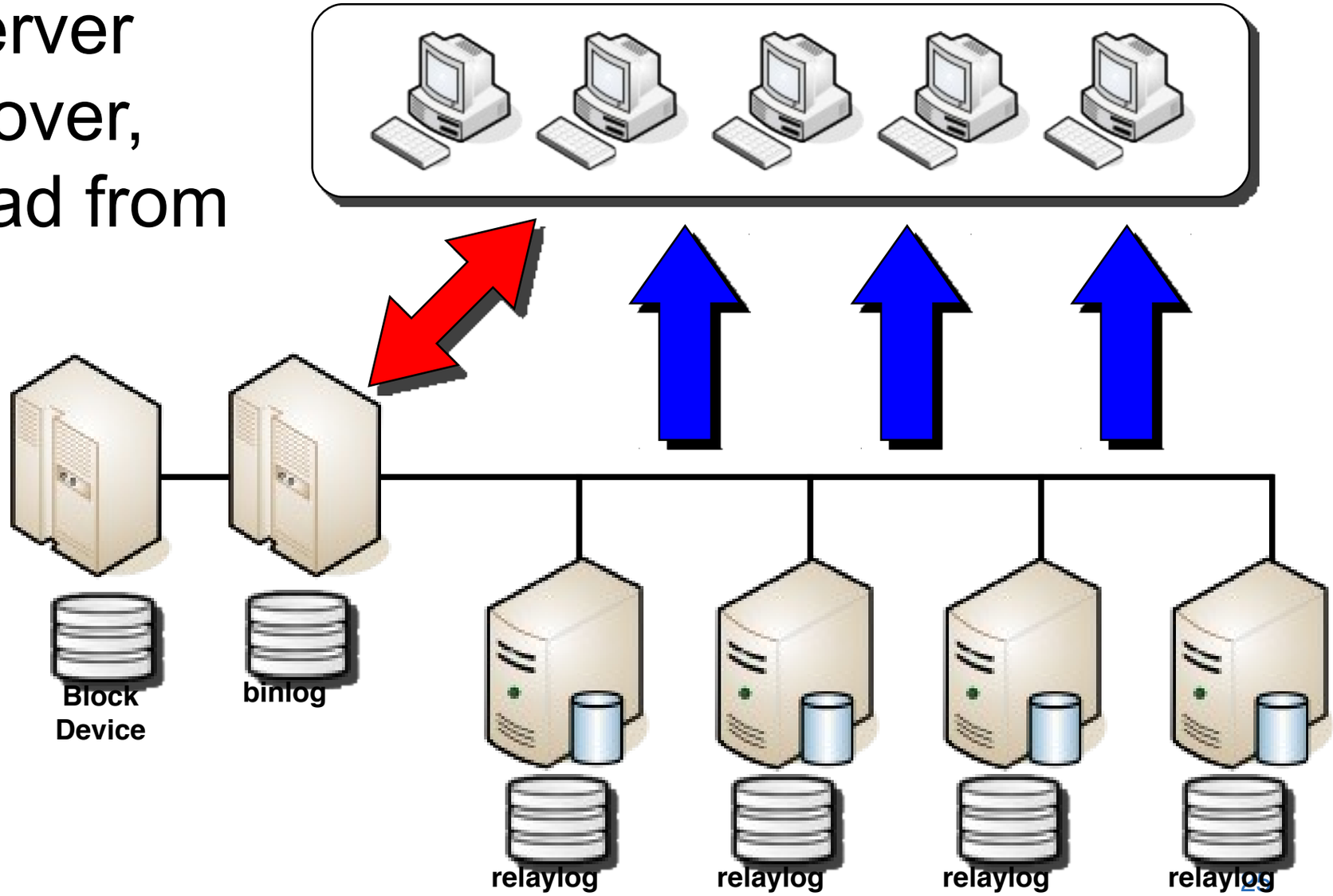
Synchronous Replication through DRBD

Other Features

- Fully certified, Linux-only environment
  - Works with Pacemaker and Heartbeat
- Works with InnoDB only
- Inexpensive solution

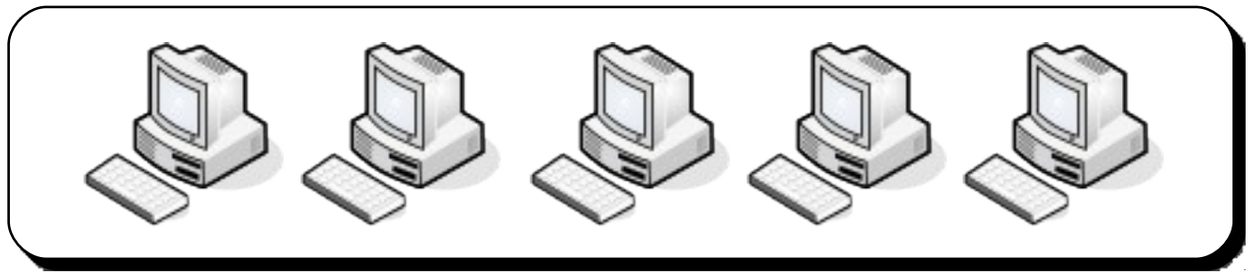
DRBD + MySQL Replication

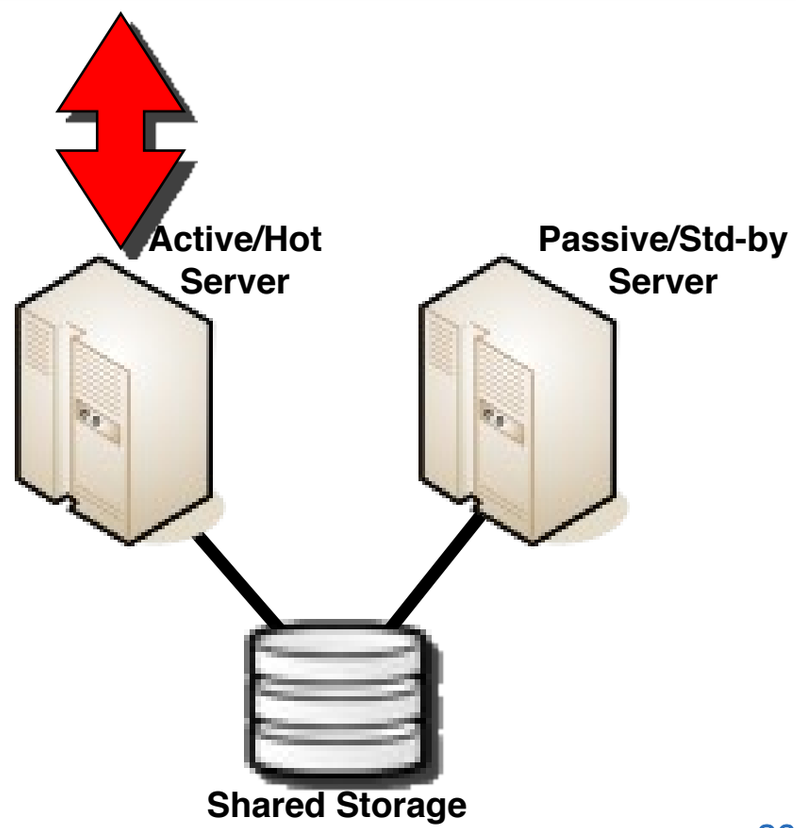
- The combination of DRBD and MySQL Replication provides more availability for scheduled and unscheduled downtime
- In case of fault of the master server, DRBD switches over to the stand-by server
- During the switchover, clients can still read from the slaves
- For planned downtime and upgrades, DBAs can apply rolling upgrades



Active/Passive Clusters through Shared Storage

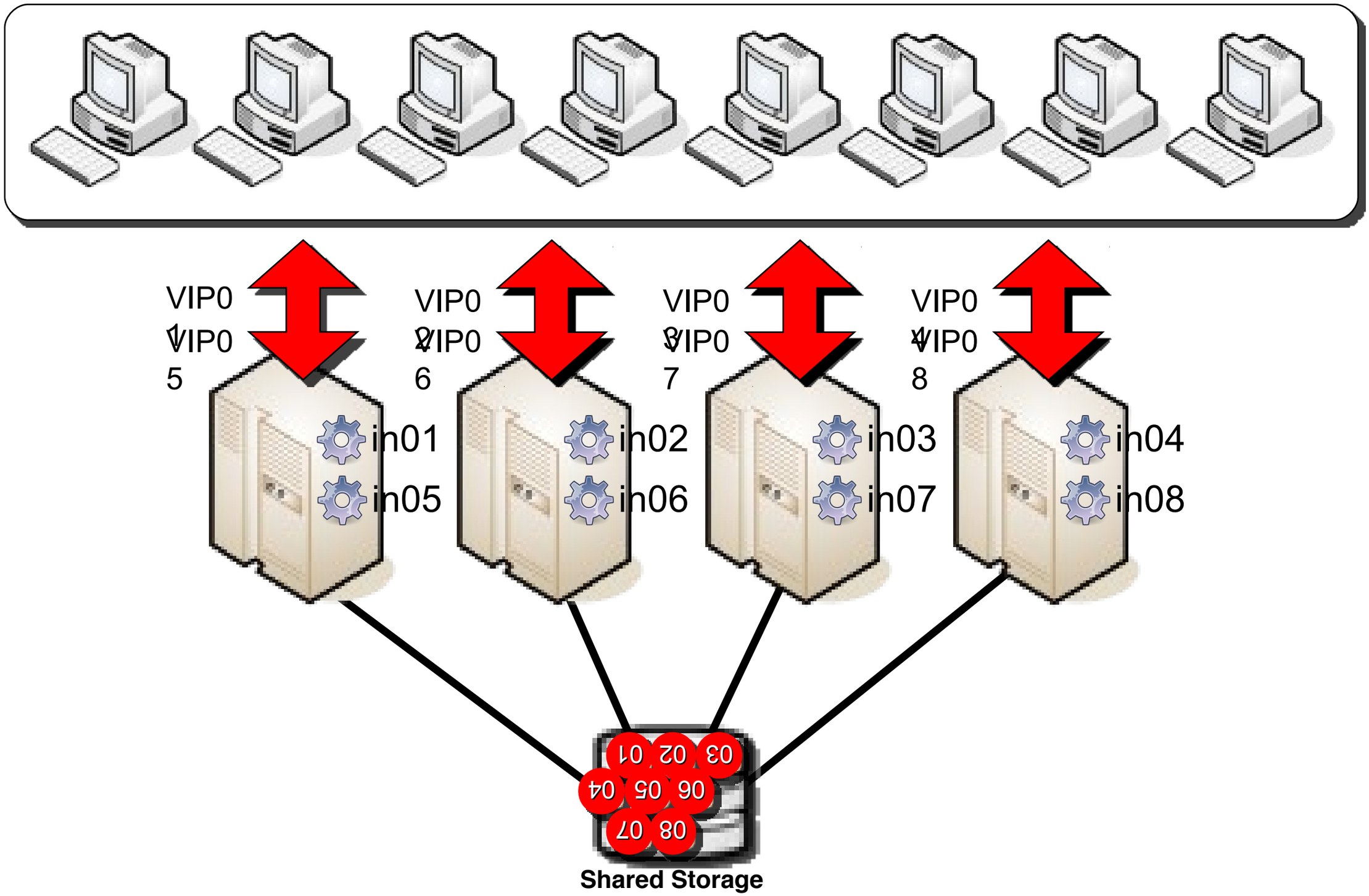
- Similar to DRBD, but data is not replicated through the network.
  - Redundancy and replication must be guaranteed by the shared storage
  - Shared storage is usually Storage Area Networks (SAN) or Network Attached Storage (NAS)
- InnoDB only, available with Linux, Windows or Solaris
- Used with FS like GFS or OCFS2
- In case of fault:
  - The master server is taken down
  - The mysqld on the stand-by server starts
  - The virtual IP address moves to the stand-by server
  - The stand-by server becomes active





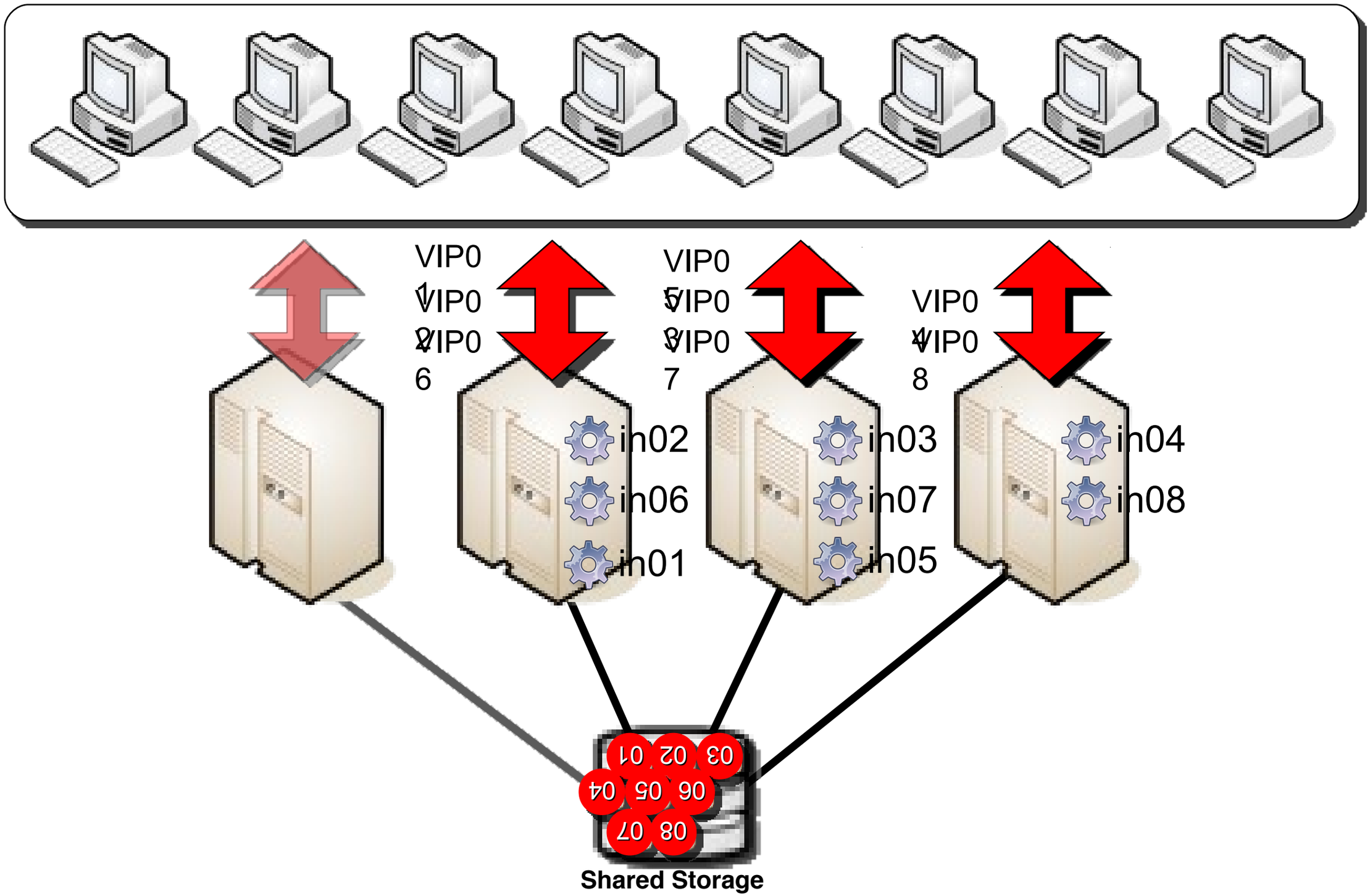
Active/Passive Clusters through Shared Storage

Large Deployments



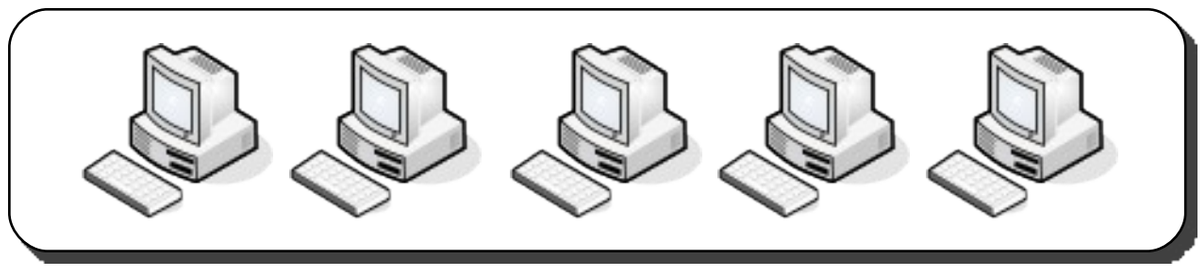
Active/Passive Clusters through Shared Storage

Failover in Large Deployments

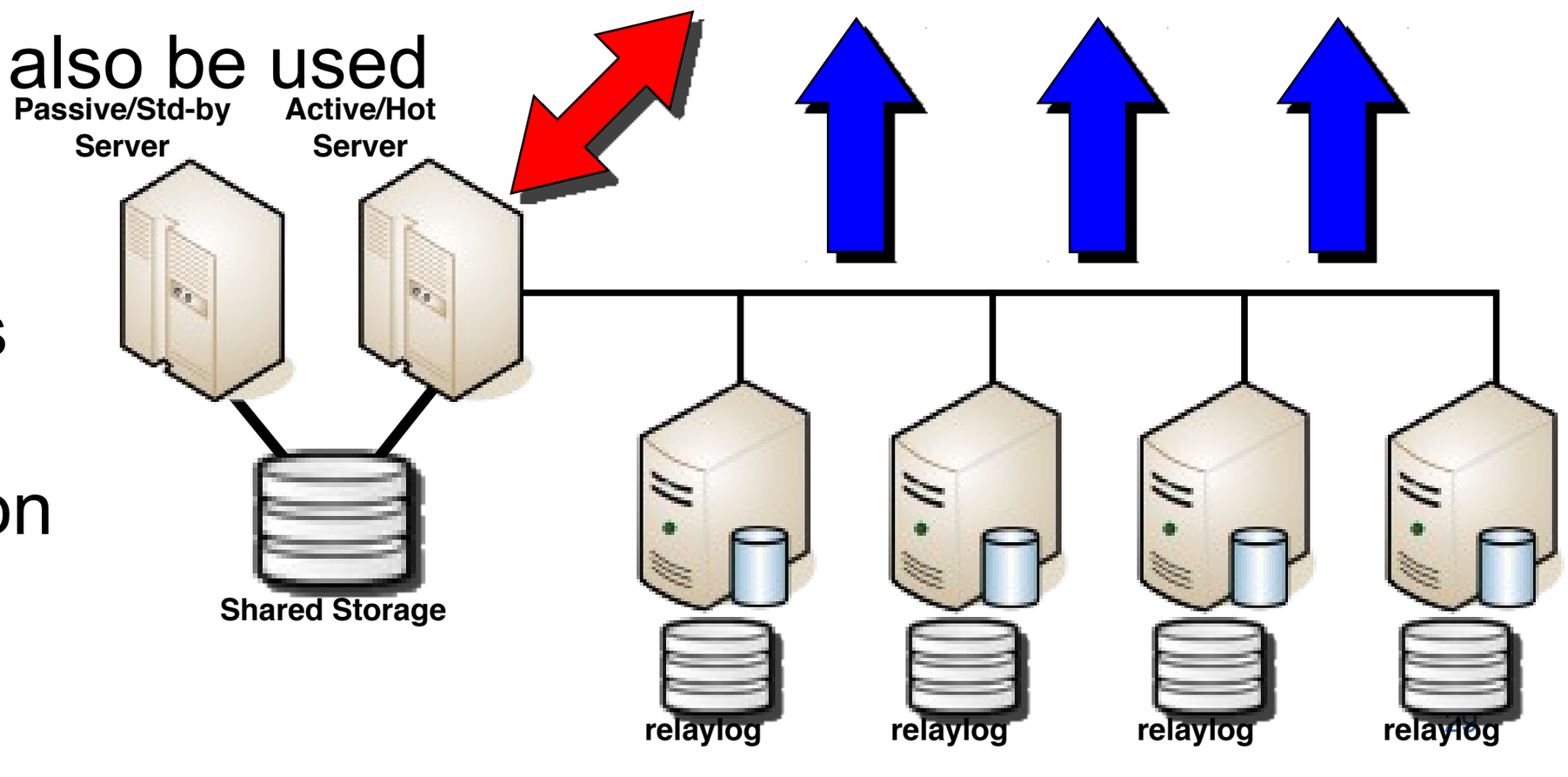


Active/Passive Clusters plus Replication

- The combination of active/passive cluster + MySQL Replication provides more availability for scheduled and unscheduled downtime
- In case of fault, the master, active server switches over to the stand-by server
- Slaves will pull data from the new slave

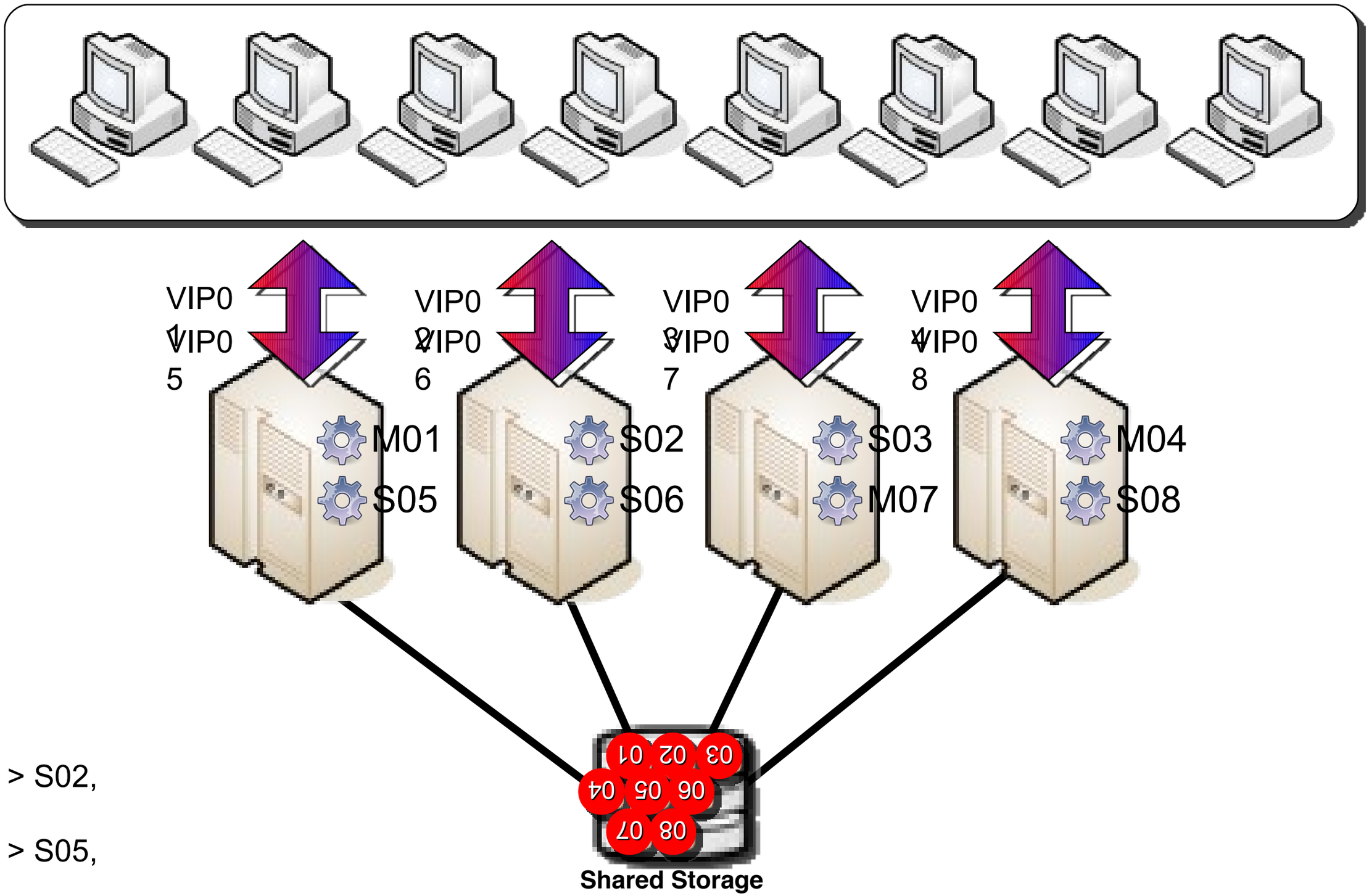


- Replication can also be used for planned downtime and rolling upgrades
- Replicated data may be stored on shared storage



Active/Passive Clusters plus Replication through Shared Storage

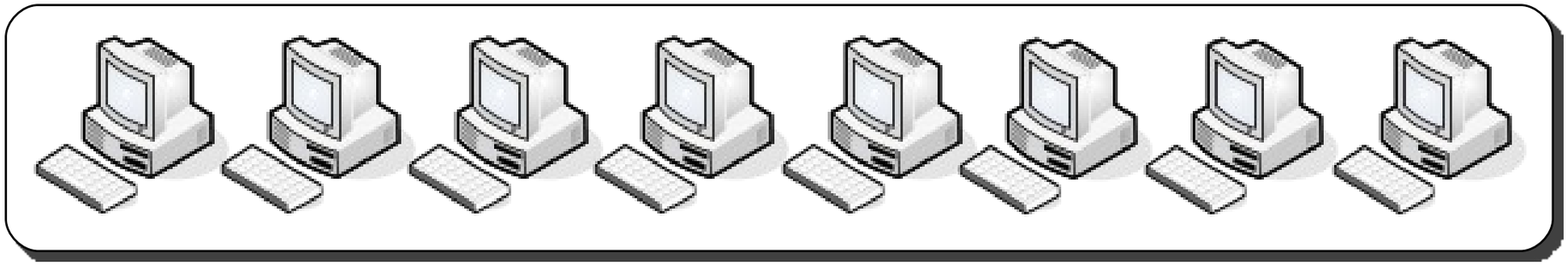
Large Deployments

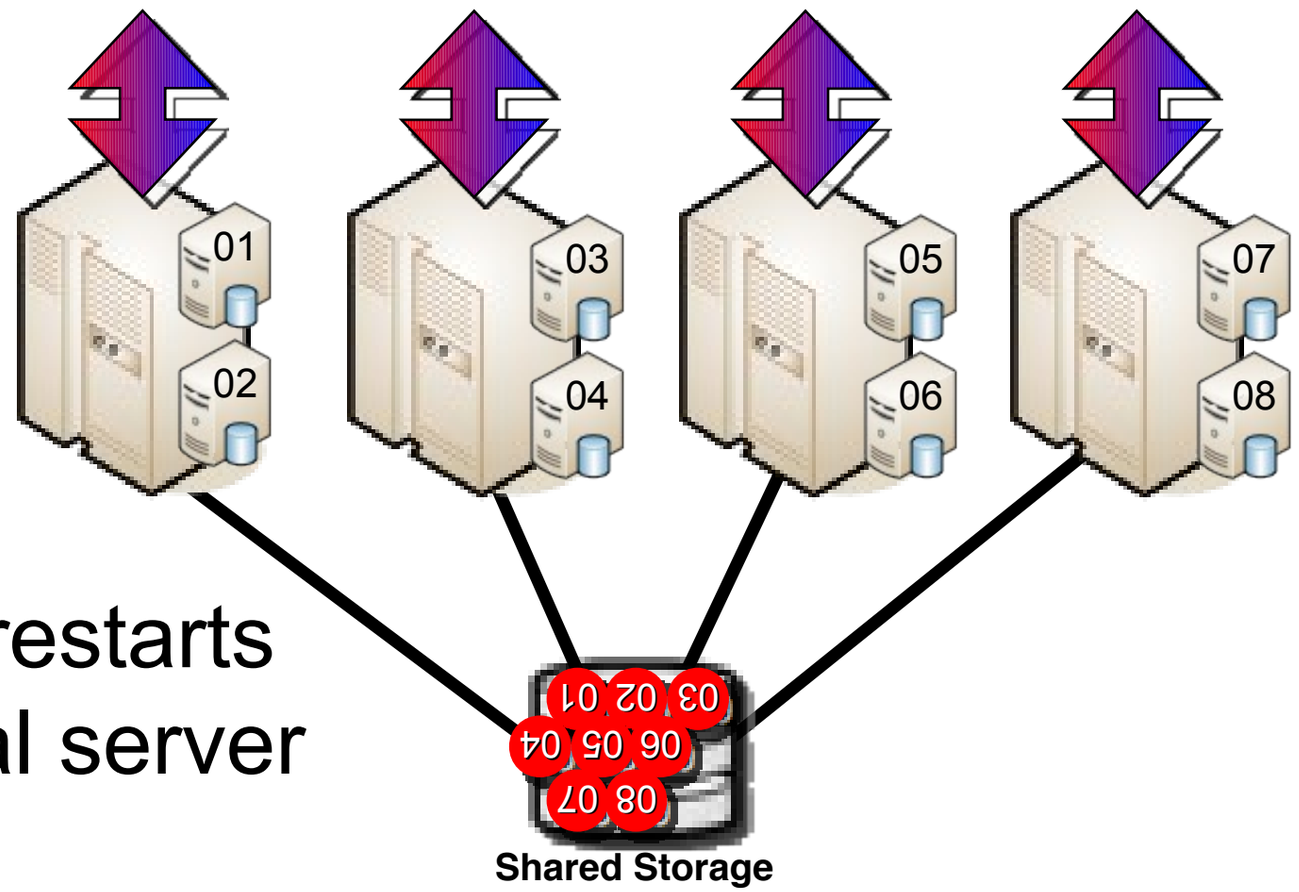


M01 > S02, S03
M04 > S05, S06

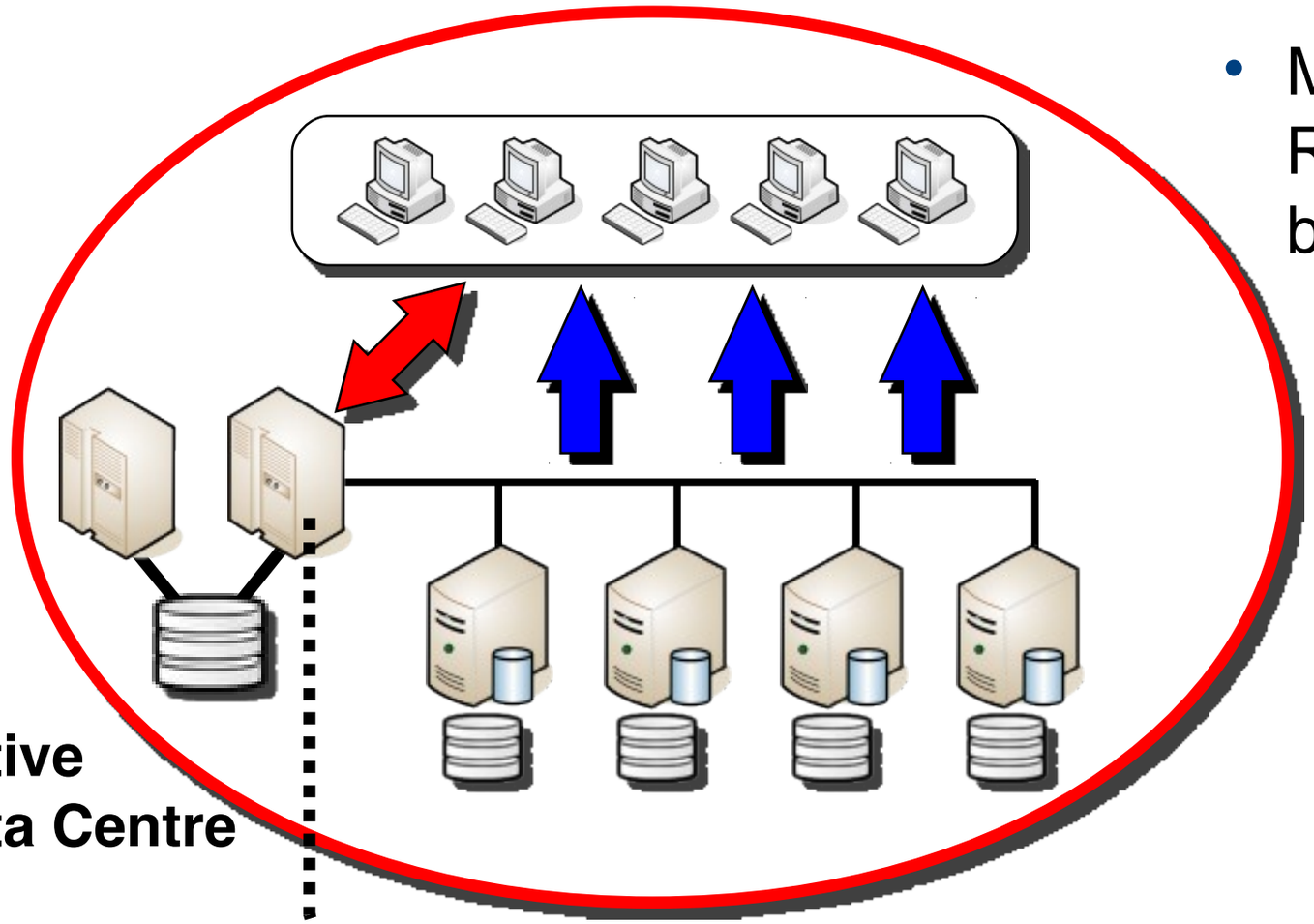
Virtualised Environments

- MySQL Servers - masters or slaves - are located on any available physical server
- High availability and load balancing are provided by the virtualised software
- Data storage is usually managed by the virtualised software but it is masked as local storage
- In case of fault, the virtualised software restarts on any other physical server
- InnoDB only

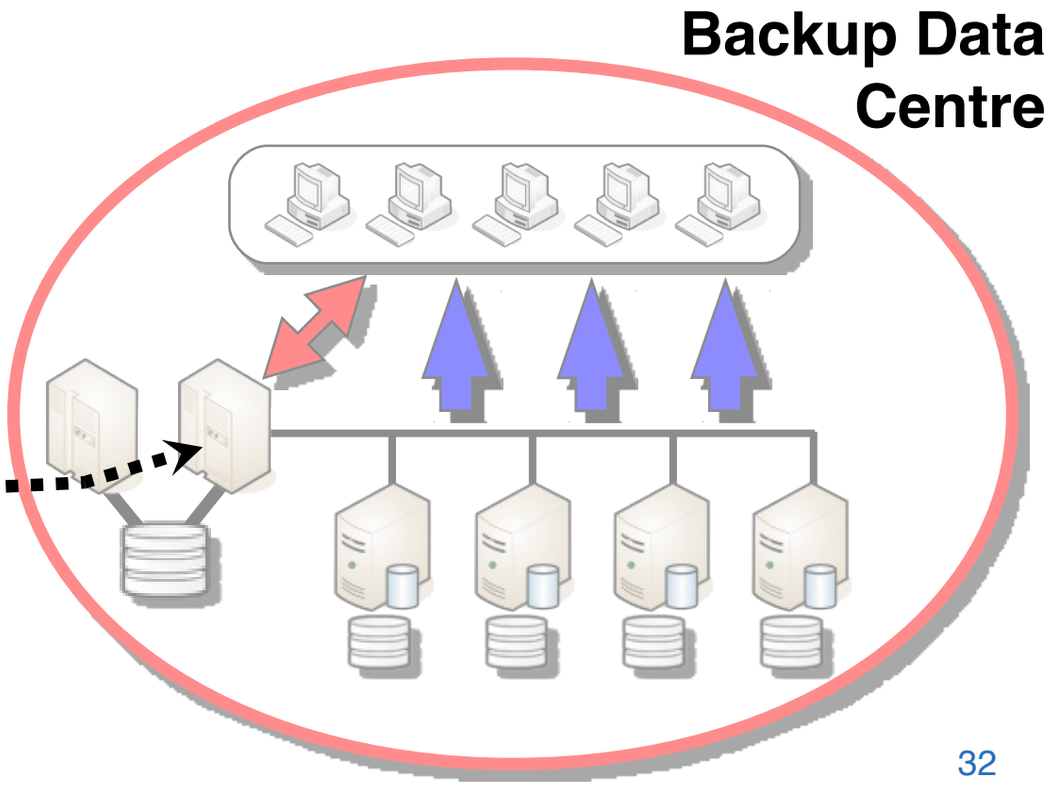




Geographical Replication for Disaster Recovery

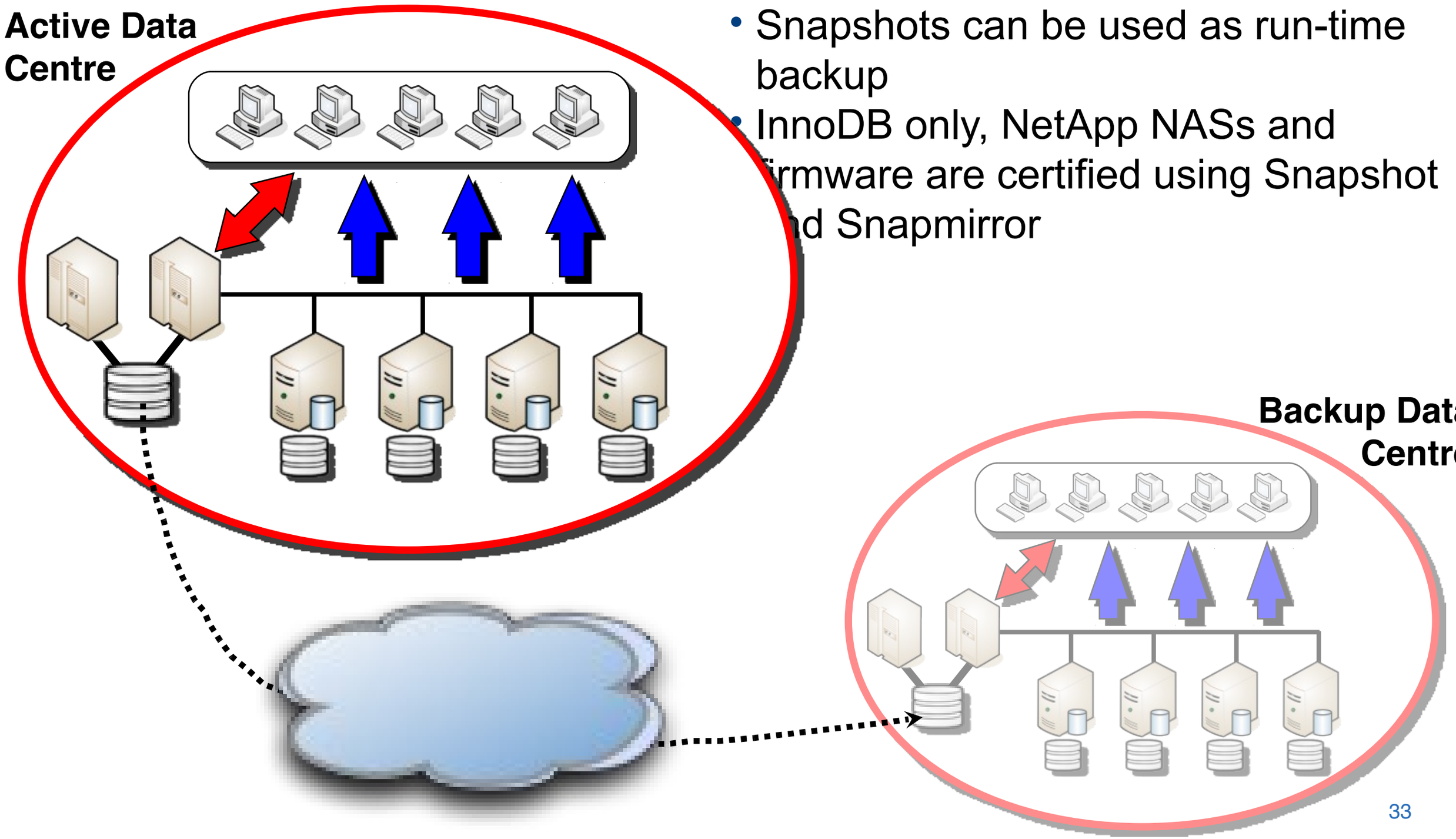


- Master-Master Asynchronous Replication is used to update the backup data centre
  - In case of fault, the network traffic is redirected to the backup data centre. Failback must be executed manually
  - Cross-platform and cross-engine



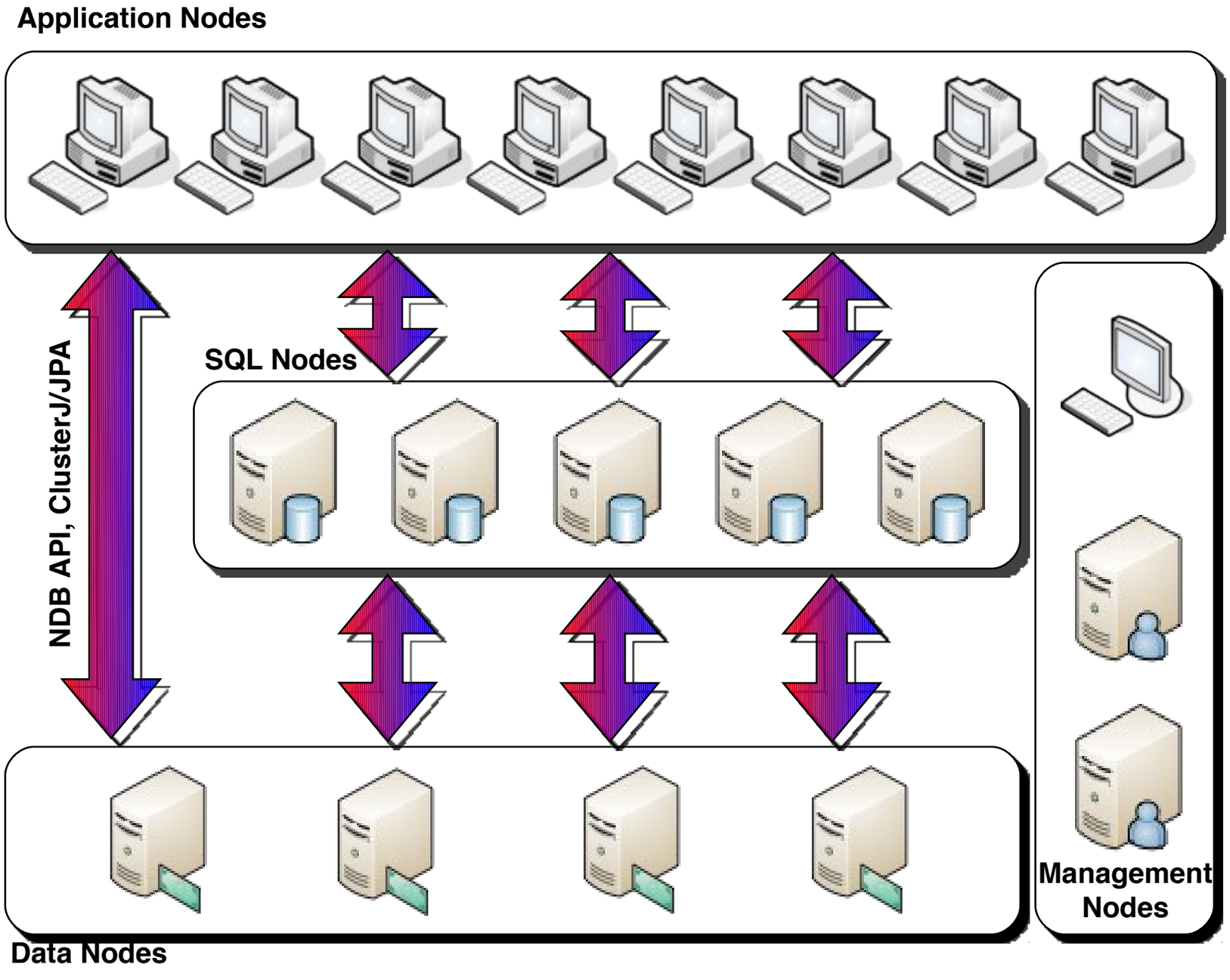
Storage Snapshots for Disaster Recovery

- Snapshots are managed by the NAS and SAN firmware. There is usually a short read-only freeze
  - Snapshots can be used as run-time backup
  - InnoDB only, NetApp NASs and firmware are certified using Snapshot and Snapmirror



MySQL Cluster

- Shared-nothing, fully transactional and distributed architecture used for high volume and small transactions.
- MySQL Cluster is based on the NDB (Network DataBase) Storage Engine
- Data is distributed for scalability and performance, and it is replicated for redundancy on multiple data nodes.
- Nodes in a cluster:
  - *SQL Nodes:* provide the SQL interface to NDB
  - *API Nodes:* provide the native NDB API
  - *Data Nodes:* store and retrieve data, manage transactions
  - *Management Nodes*: manage the Cluster
- Load balanced
- Memory or disk-based
- Geographically replicated for disaster recovery
- Full online operation for maintenance and administration



Thank You!

Joffrey MICHAÏE
Consultant MySQL
joffrey@skysql.com