Shadowbase®

GRAVIC

VNUG
VIKING NONSTOP USER GROUP

# The Evolution of Async and Sync Replication for High and Continuous Availability NonStop-Based Architectures
## VNUG: 26-27 May 2010


**Paul J. Holenstein**

**Executive Vice President**

**Gravic, Inc.**

# Agenda

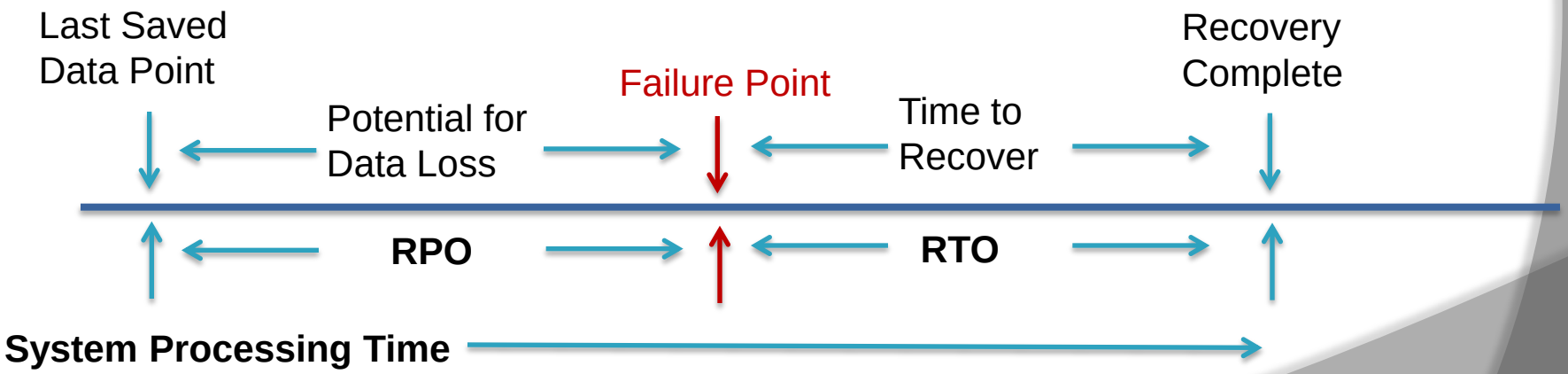
- Business Continuity Overview
- Replication Technologies
  - Async and Sync
- Replication Architectures
  - High Availability: Active/Passive
  - Better Availability: Sizzling Hot Takeover Active/“Almost Active”
  - Continuous Availability: Active/Active
- Issues to Consider When Picking a BC Architecture
- Business Continuity Case Studies
- References for More Information

**Questions? Please ask as we go along.**

# Business Continuity Overview

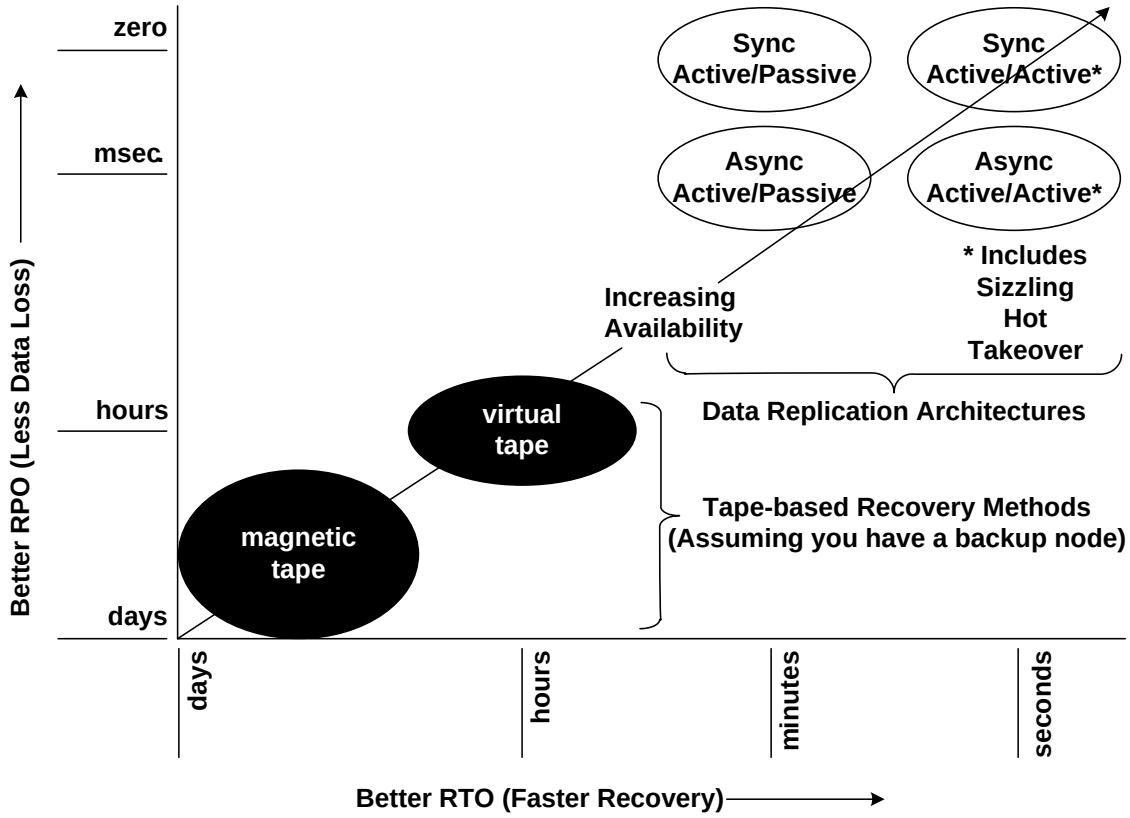
- **Recovery Time and Recovery Point Objectives**
  - Recovery Time Objective (RTO) and Recovery Point Objective (RPO) are two commonly used terms to describe business continuity requirements.
    - → RPO – describes the point in time to which the data must be recovered
    - → RTO – describes the time from when a failure occurs until the business process must become active again.

## RPO/RTO Relationship



# Business Continuity Overview

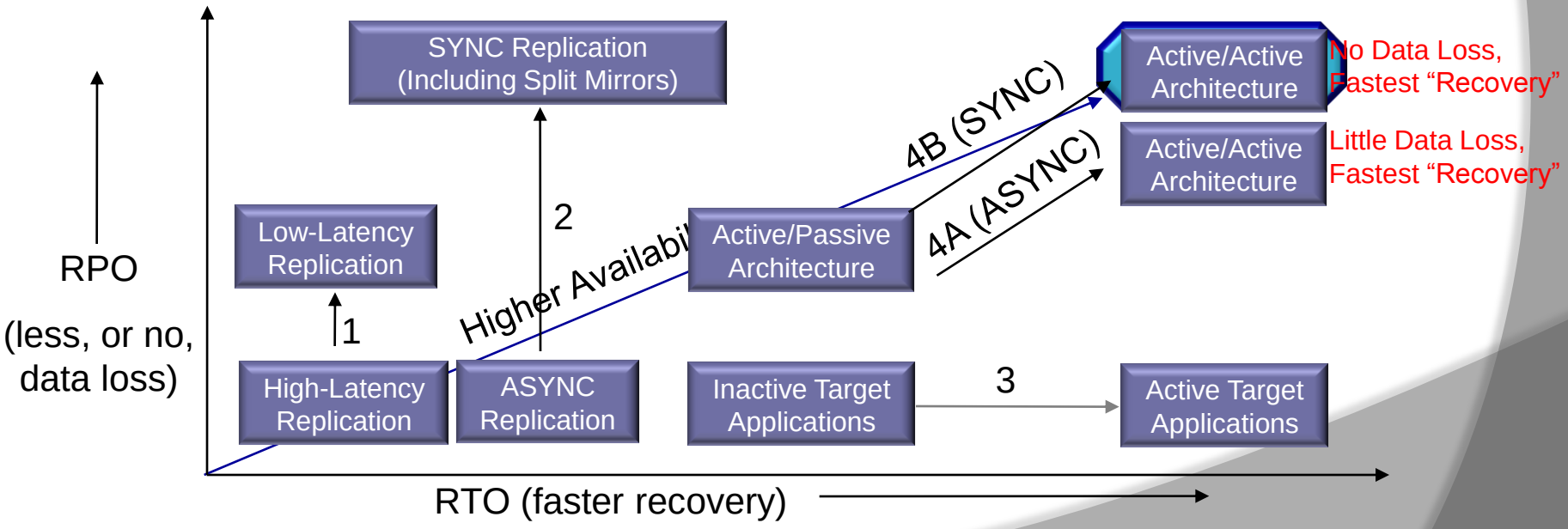
## Business Continuity Continuum



# Business Continuity Overview

## Replication Characteristics Affect RPO/RTO Levels that can be Attained

1. Replication Latency – affects RPO (amount of data loss at failure)
2. Async vs Sync Replication – improves RPO, but adds "Application Latency"
3. Target Applications Active – improves RTO and usability of target DB (e.g. queries)
4. Active/Passive vs Active/Active Architecture – dramatically improves RTO & RPO



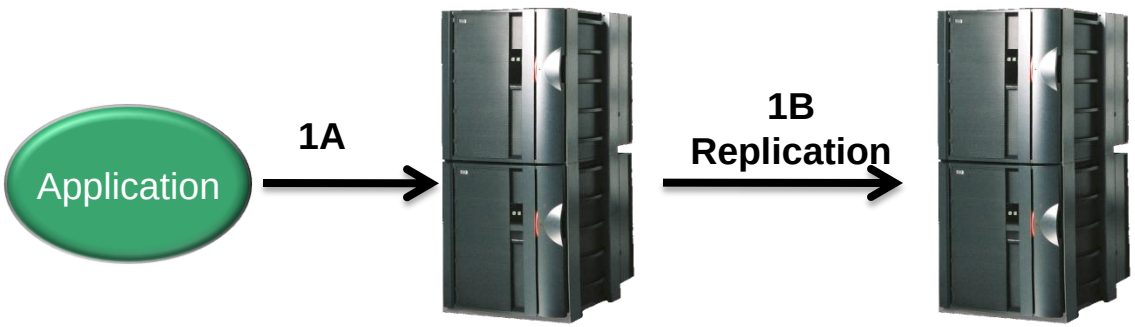
# Replication Architectures

## Asynchronous Replication (Current Technology)

- Replication decoupled from the application processing
  - Runs independently of application

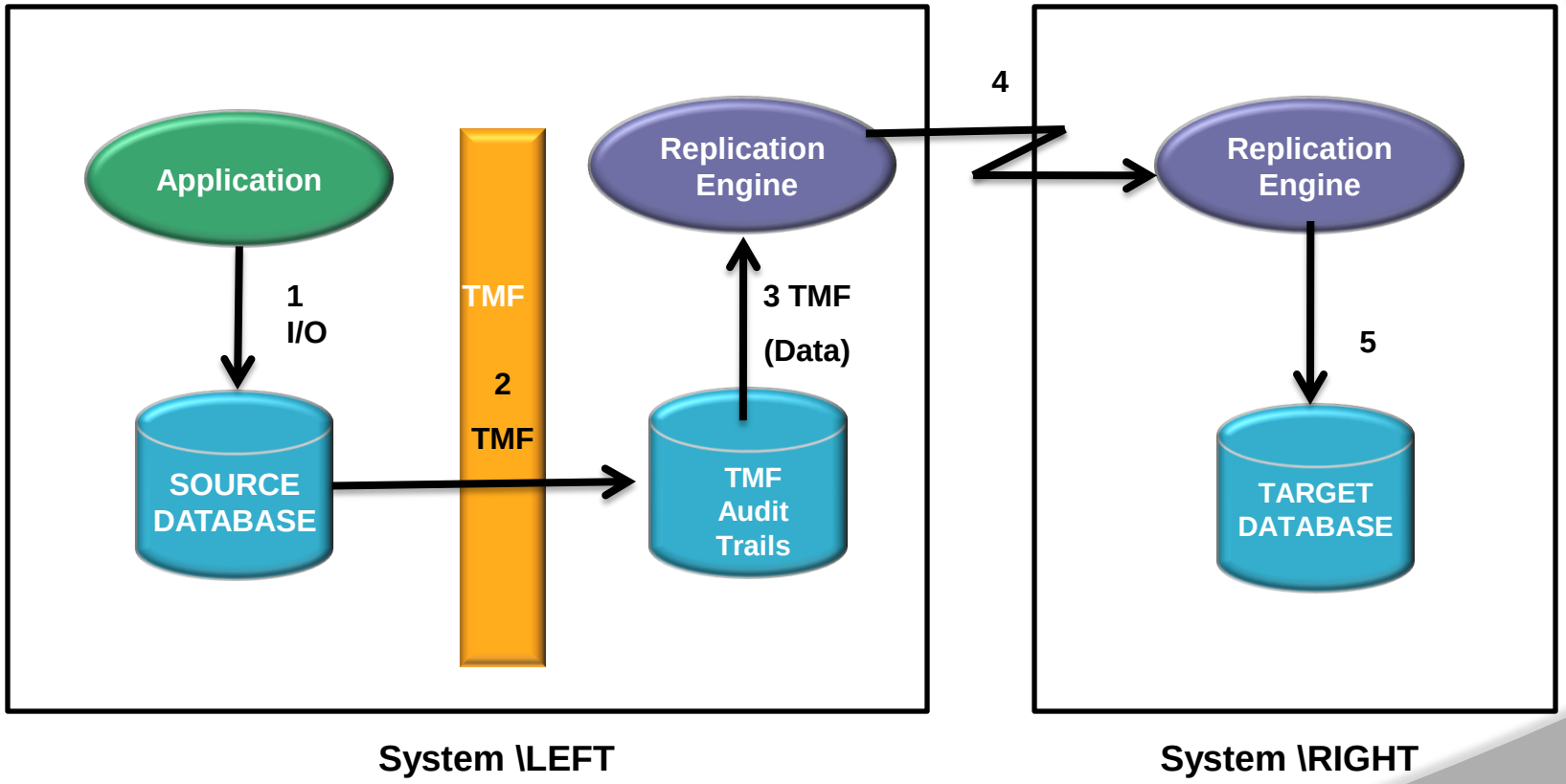
## Issues

- *Replication Latency* determines data loss at failure (RPO)
  - Need *low latency* data replication engine to minimize data loss at failure



# Replication Architectures

## Uni-Dir Asynchronous Replication (Current Technology)



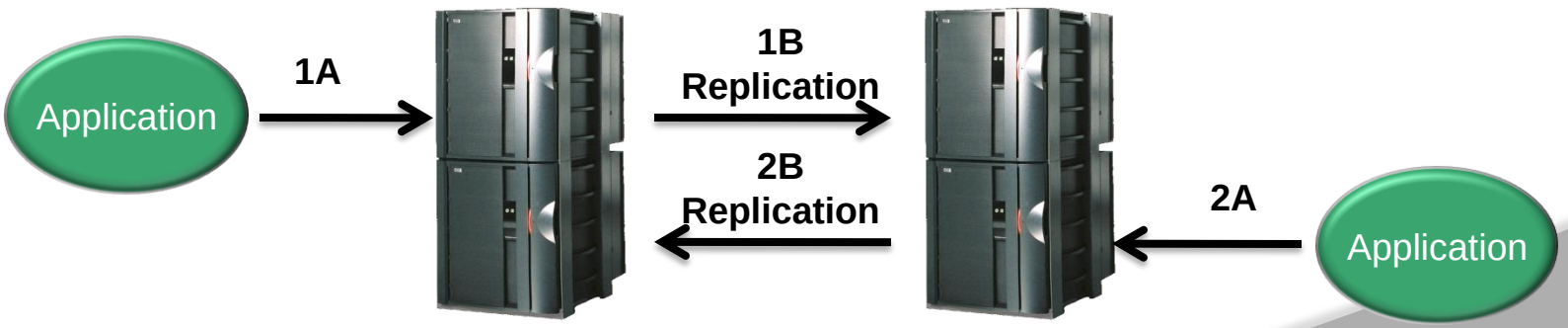
# Replication Architectures

## Asynchronous Replication (Current Technology)

- Replication decoupled from the application processing
  - Runs independently of application

## Issues

- *Replication Latency* determines data loss at failure (RPO)
  - Need *low latency* data replication engine to minimize data loss at failure
- If Active/Active:
  - Data Collisions can occur (during Replication Latency window)
  - Need special architectures/algorithms to avoid vs. identify & resolve (application issue, see Case Studies)



# Replication Engine Technology

## Async Replication - How are Data Collisions Identified?


# Replication Engine Technology

## Async Replication - How are Data Collisions Identified?

- Send Source's *Before* as well as *After* images to target
- Compare Source's *Before Image* to target's *Disk Image* to check

| Source Event Type | Target Disk Image |
|---|---|
| INSERT | Does Not Exist – Apply Insert<br>Exists – Collision |
| UPDATE | Exists & Same – Apply Update<br>Does Not Exist – Collision<br>Exists & Different – Collision |
| DELETE | Exists & Same – Apply Delete<br>Does Not Exist – Collision<br>Exists & Different – Collision |

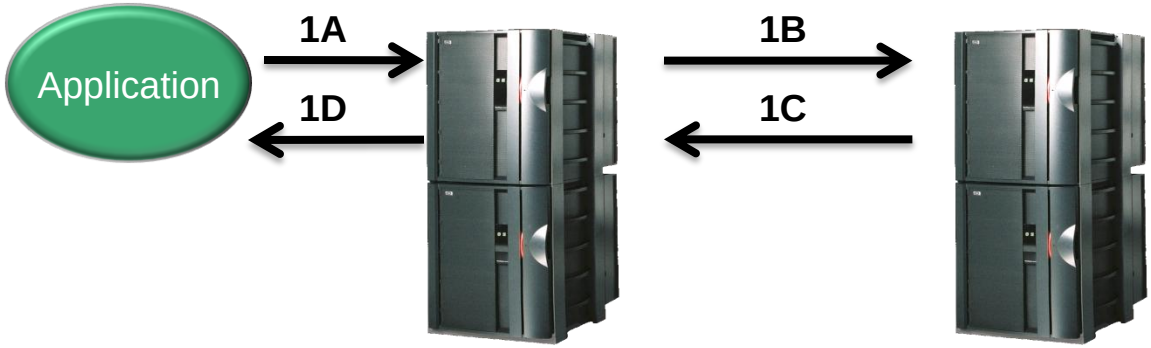
# Replication Architectures

## Synchronous Replication (Future Technology)

- No (committed) data loss at failure of a node (RPO = 0)

## Issues

- Replication is part of the source application's transaction
  - Adds *Application Latency,* effectively *slowing* the application's transactions
- What to do if network or target system is down?
  - So-called *Split Brain* syndrome (next slide set…)



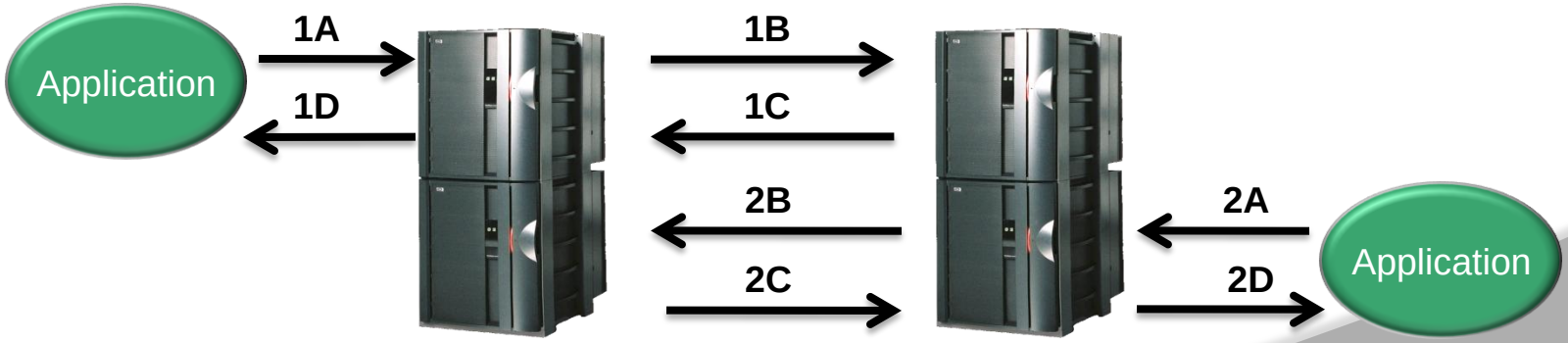
# Replication Architectures

## Synchronous Replication (Future Technology)

- No (committed) data loss at failure of a node (RPO = 0)
- If Active/Active:
  - Data collisions are avoided (but become tx *timeouts;* back off & resubmit tx)

## Issues

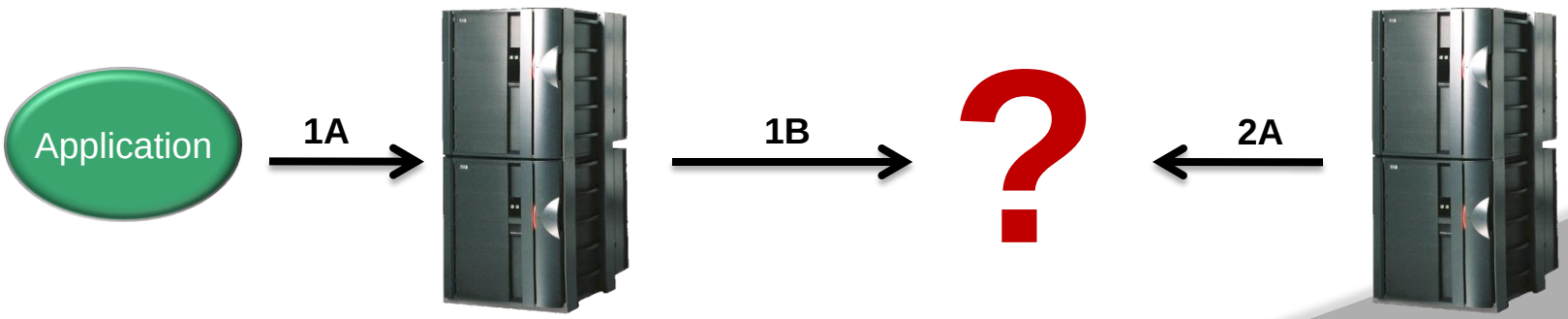
- Replication is part of the source application's transaction
  - Adds *Application Latency,* effectively *slowing* the application's transactions
- What to do if network or target system is down?
  - So-called *Split Brain* syndrome (next slide set…)



# Replication Architectures

## Synchronous Replication – Split Brain Defined

- *Split Brain* occurs when one (or more) nodes do not know the status/state of the other node(s) in the network
- Typically only an issue in these cases:
  - Zero Data Loss is required (e.g. regulations)
  - Running Bi-Dir (Active/Active) and Data Collisions are possible



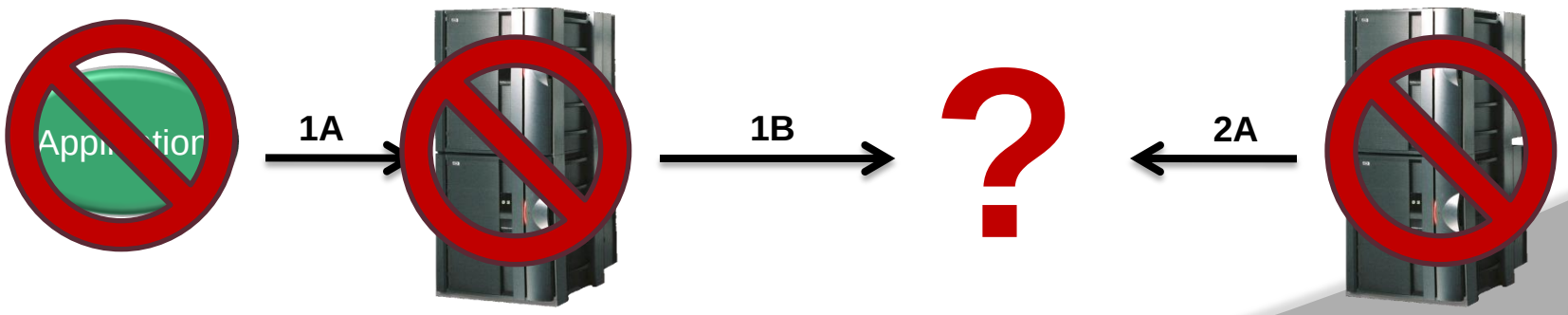
# Replication Architectures

## Synchronous Replication – Split Brain Defined

- Split Brain occurs when one (or more) nodes do not know the status/state of the other node(s) in the network

## Split Brain Resolution Options

- **1) Fail all nodes:** fails application - reduces application availability



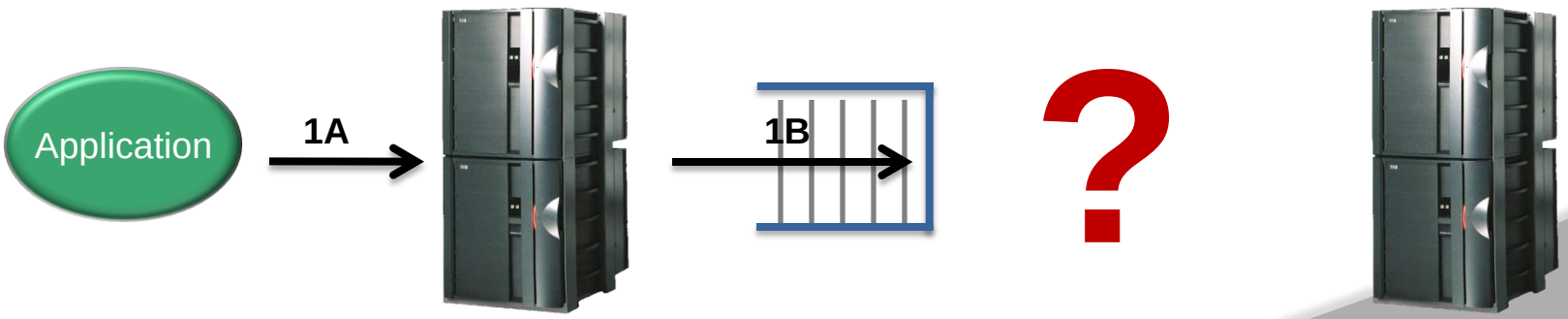
# Replication Architectures

## Synchronous Replication – Split Brain Defined

- Split Brain occurs when one (or more) nodes do not know the status/state of the other node(s) in the network

## Split Brain Resolution Options

- 1) Fail all nodes
- **2) Fail all nodes but one:** Uni-Dir No Change (queue changes)



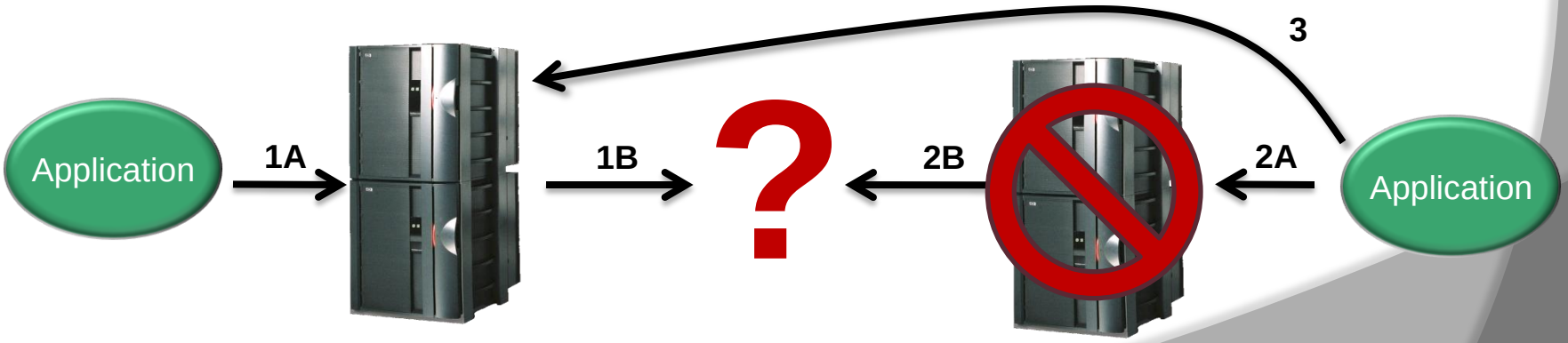
# Replication Architectures

## Synchronous Replication – Split Brain Defined

- Split Brain occurs when one (or more) nodes do not know the status/state of the other node(s) in the network

## Split Brain Resolution Options

- 1) Fail all nodes
- **2) Fail all nodes but one:** Bi-Dir (failover users to avoid Data Collisions)



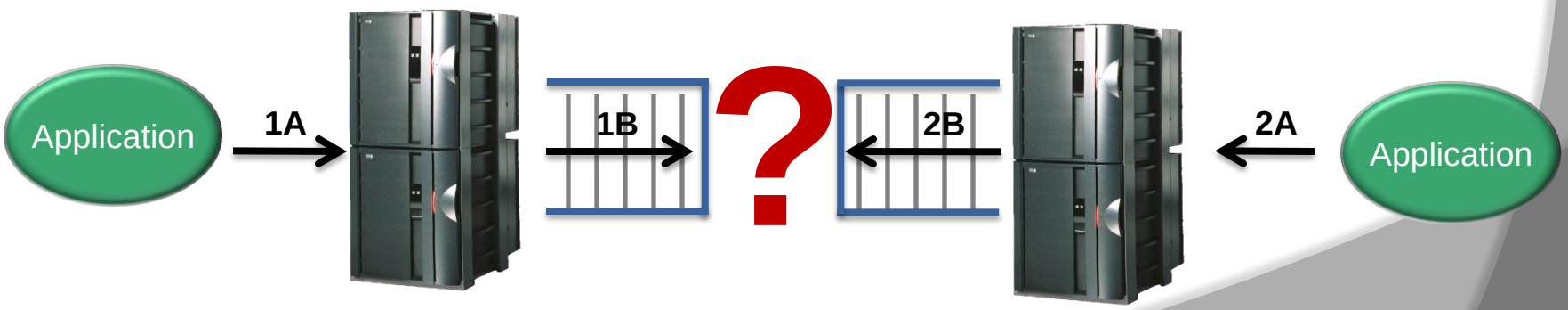
# Replication Architectures

## Synchronous Replication – Split Brain Defined

- Split Brain occurs when one (or more) nodes do not know the status/state of the other node(s) in the network

## Split Brain Resolution Options

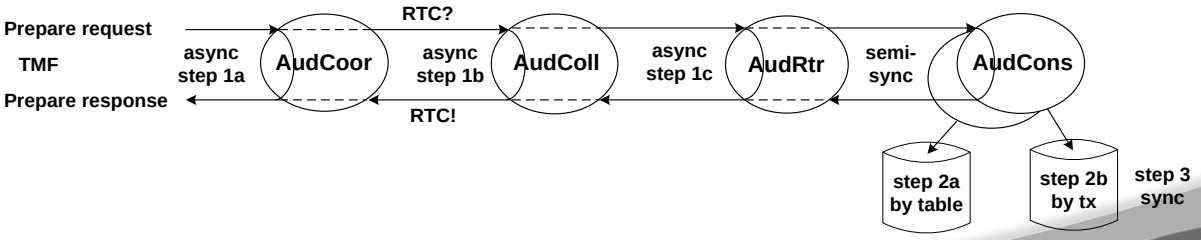
- 1) Fail all nodes
- 2) Fail all nodes but one
- **3) Allow nodes to run independently:** Bi-Dir both queue



# Replication Architectures

## Synchronous Replication – Recovery Methods

- Recovery modes 2 & 3 require *Async Escalation Recovery Mode*
- This consists of escalation of replication capabilities to reach end state:
  - Start in “Async Queue Mode” until problem is repaired;
  - Then enter “Async Replication Mode” until queue drains;
  - Then enter “’Semi-Sync’ Mode” until all Async tx’s end;
  - Then enter “Full Sync Mode” when all Semi-Sync tx’s end…
  - …Any problems, start over…
- Refer to September/October 2009 Availability Corner *Connection* Article
  - **“Part 18: Recovering from Synchronous Replication Failures”**



# Replication Architectures

## Synchronous Replication – Contrast 2 *Transactional* Methods

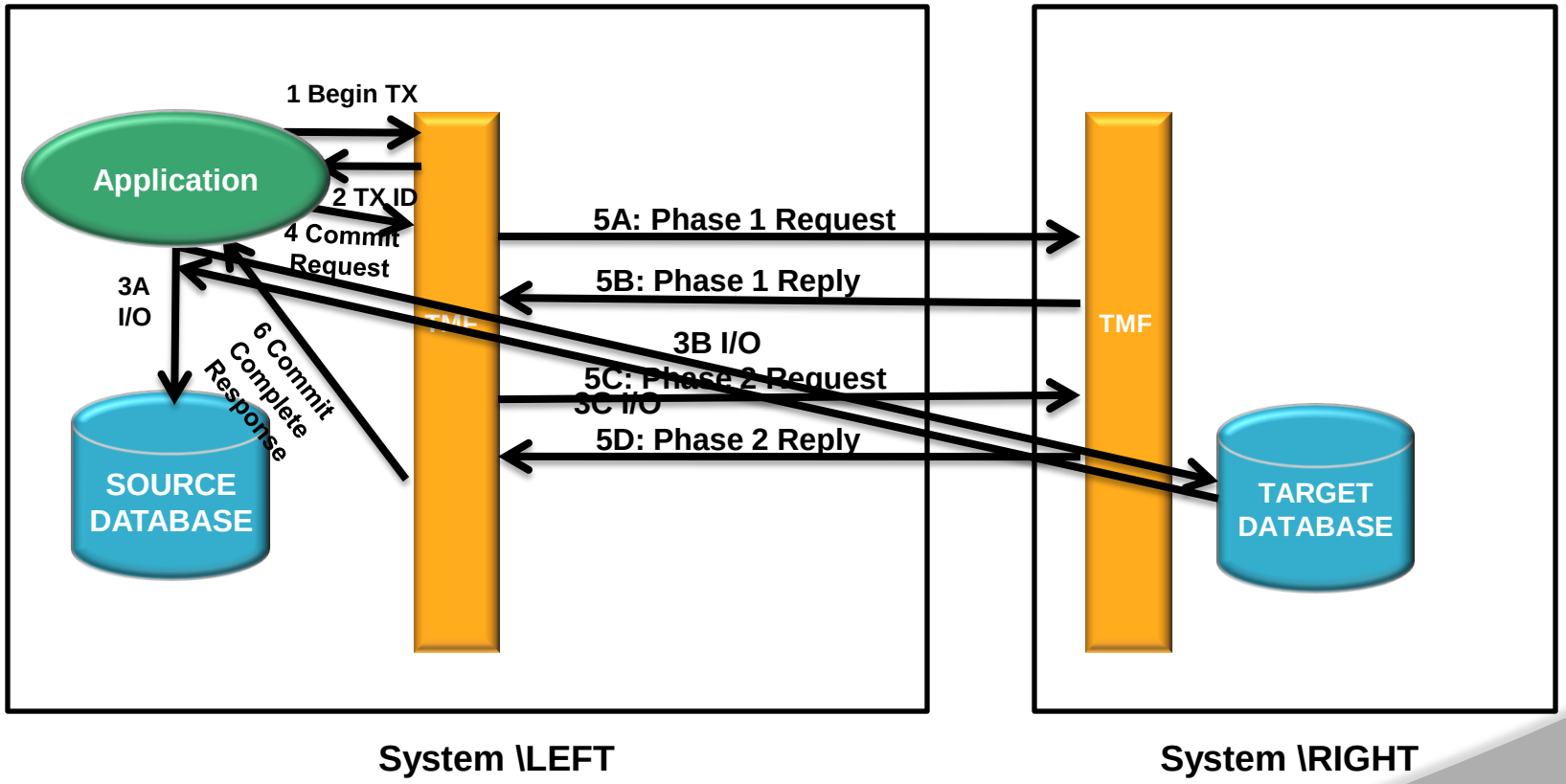
- There are more, we'll just discuss two today
  - "*Dual Writes*" (DW), also called "*Network Transactions"*
  - "*Coordinated Commits*" (CC)

## Dual Writes (DW), Also Called "Network Transactions"

- No replication engine used, has existed "forever"
- Application starts a local transaction
  - Escalates to a "distributed", or network, transaction when remote I/O is sent/applied
- Application does local and remote I/O directly
  - Application or library or interface process infrastructure needed; modify application?
  - Each I/O is 1 or 2 network messages to and from remote system
    - Read/Lock & Update vs. standalone Update
    - Application delayed for each round trip
  - Lots of small network messages sent/received across network
- Application commit is 2 round trips across network (2PC)
  - Application's commit call is delayed until this completes

# Replication Architectures

## Uni-Dir Synchronous Replication (Dual Writes/Network Transactions)



# Replication Architectures

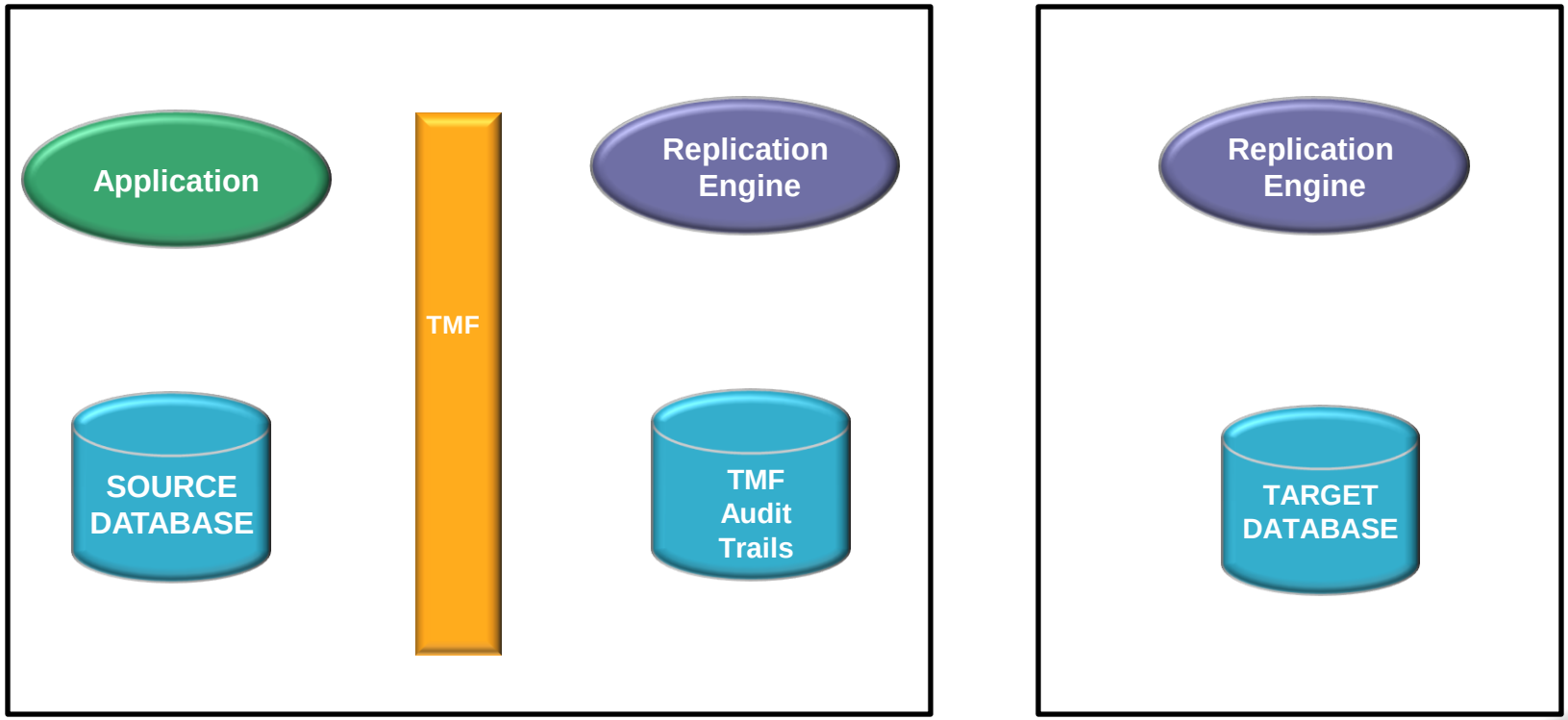
## Synchronous Replication – Contrast 2 *Transactional* Methods

### Coordinated Commits (CC)

- (Async) replication engine used
- Application starts a local transaction, it stays local
  - Replication engine “joins” it and can vote on outcome (to commit or abort)
  - Remote replication engine process starts a local transaction there as well
- Application does local I/O
  - Runs at full speed (until commit time)
  - Replication engine extracts, batches, sends, and applies I/O events to target
    - All database events for all applications batched and sent together
    - Fewer larger messages sent across network
- Application commit “decision” is 1 round trip across network
  - Replication engine checks if remote replay is ready to commit
  - Votes Yes (commit) or No (abort)
  - Application continues
  - TX Result is then replicated by replication engine and applied at target

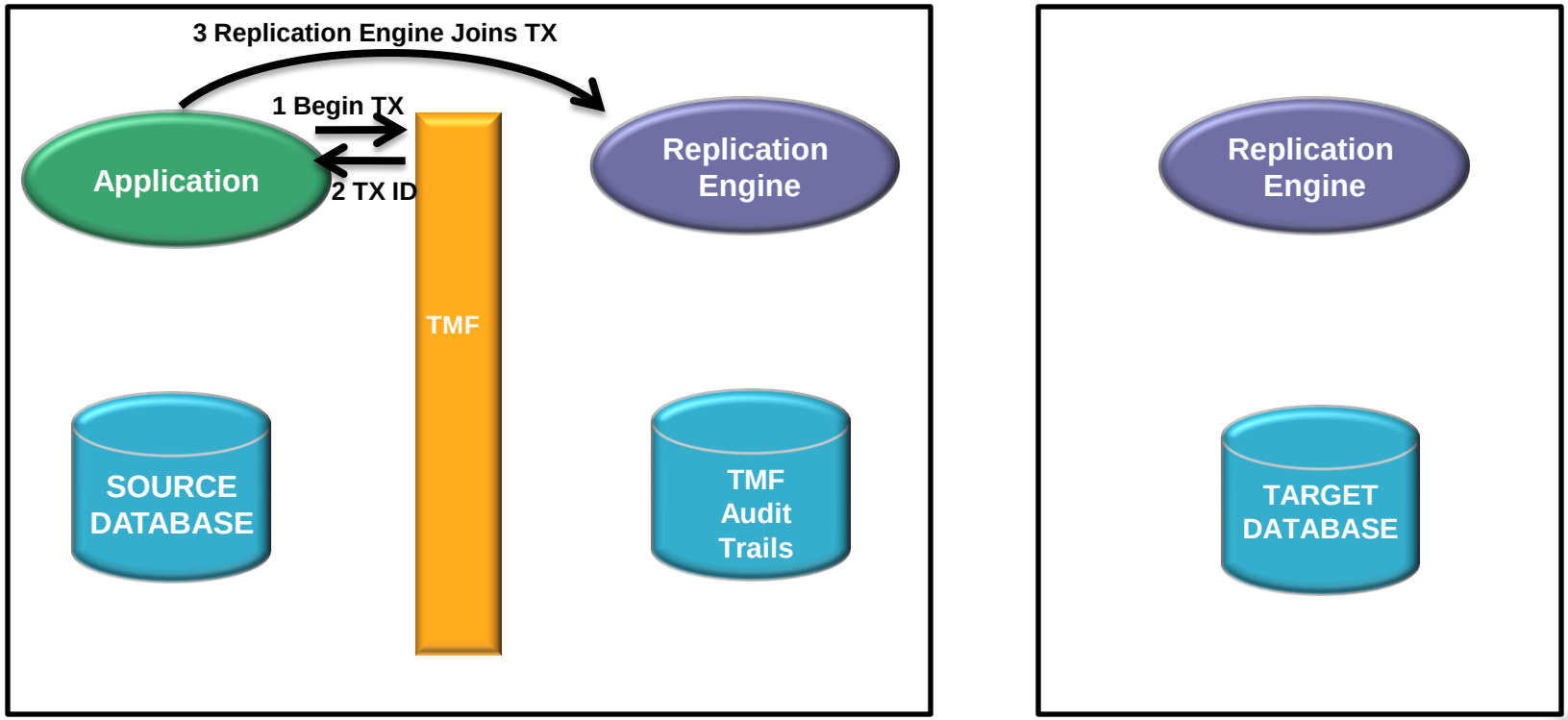
# Replication Architectures

## Uni-Dir Synchronous Replication (*Coordinated Commits*)



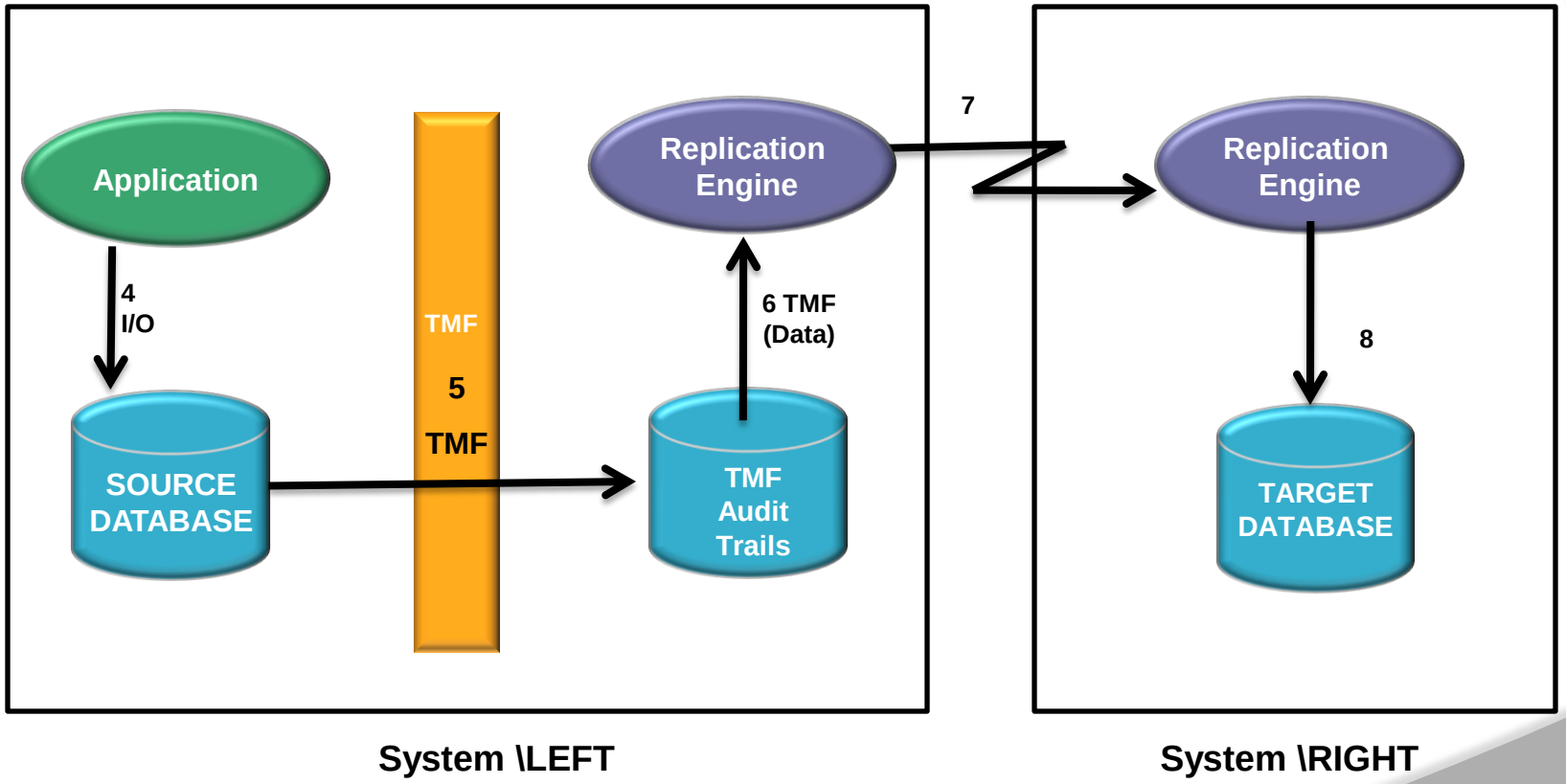
# Replication Architectures

## Uni-Dir Synchronous Replication (*Coordinated Commits*)



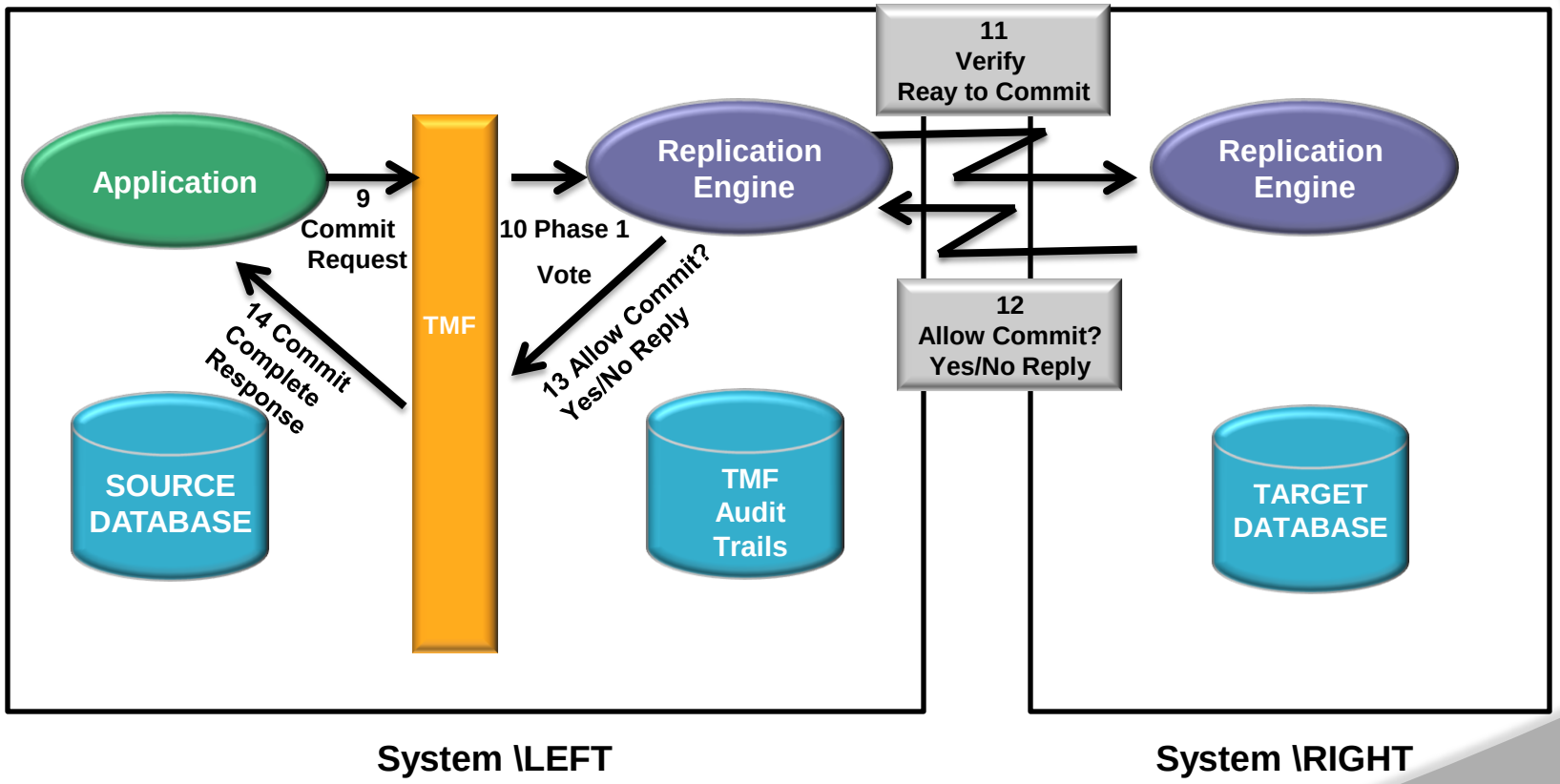
# Replication Architectures

## Uni-Dir Synchronous Replication (*Coordinated Commits*)



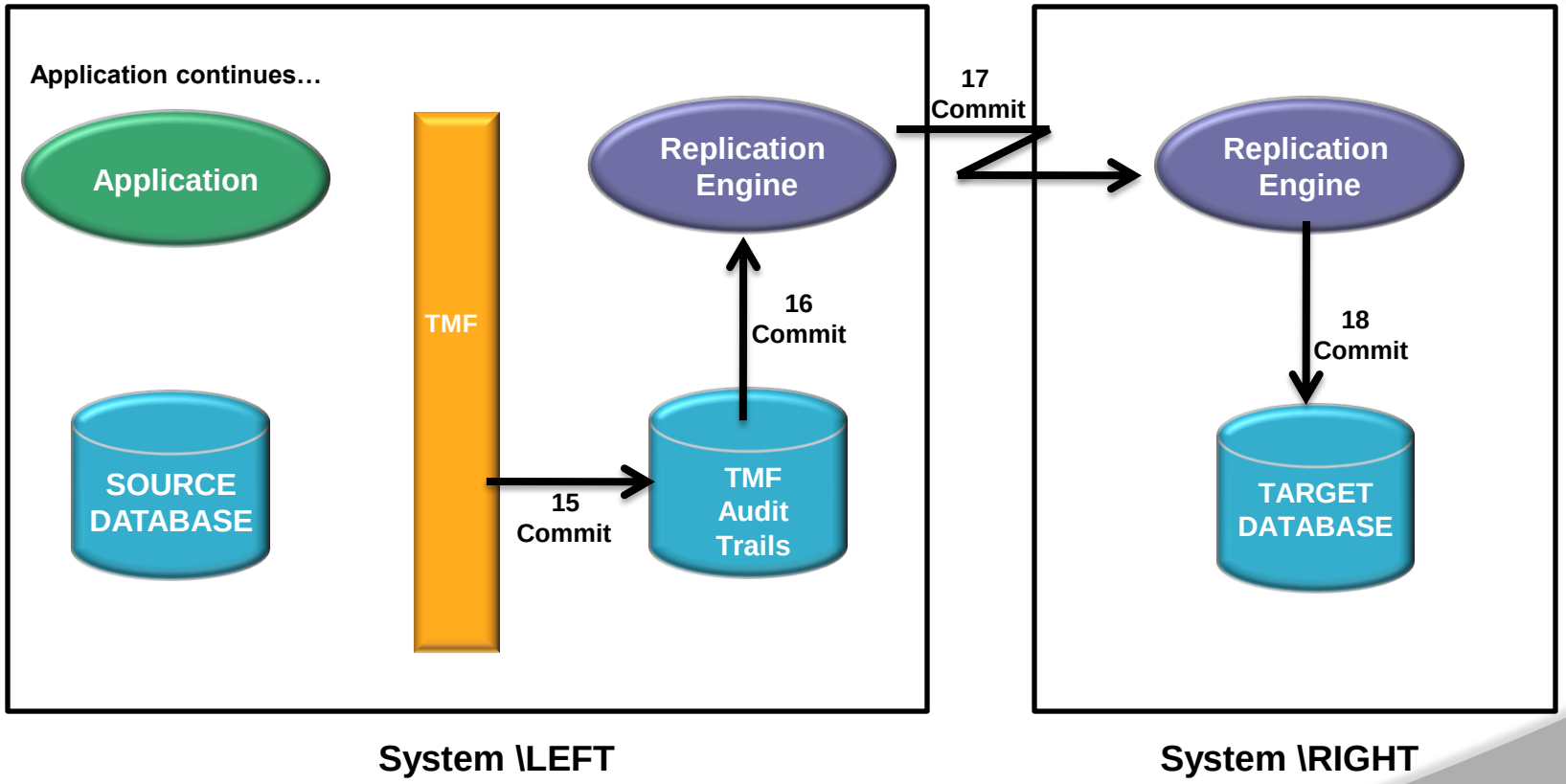
# Replication Architectures

## Uni-Dir Synchronous Replication (*Coordinated Commits*)



# Replication Architectures

## Uni-Dir Synchronous Replication (*Coordinated Commits*)




**NOTE: Coordinated Commits is a future technology and requires HP's *NonStop Synchronous Replication Gateway***

# Async and Sync Replication Summary

## CC Sync Replication Has Substantially the Same Characteristics as Async Replication Except for ***Application Latency***:

- Its ***throughput*** is substantially the same
  - May need to run more copies of the application server classes though
- Its ***failure characteristics*** are the same
  - In all modes except "any fault fails all nodes"
- Its ***I/O loading*** (for replay) is substantially the same
- Its ***network loading*** is substantially the same
- Its use for ***eliminating planned downtime (ZDM)*** is the same

## Its Main Downside is ***Application Latency***:

- This delays the source application's transaction while replication occurs.
- It will increase the number of simultaneous transactions on each node.
- It will extend the time that source tx's (and data locks) are held.
- It may increase the number of source transactions that are aborted and need to be resubmitted (e.g. if target I/O's cannot be applied or "wait" too long/timeout).

## And, Failover/Recovery will be More Complex:

- Sync Mode →Async Mode → Sync Mode.
- But this should be handled automatically by the replication engine.

# Async and Sync Replication Summary

## Synchronous Replication Benefits:

- Avoiding data loss following a node failure (RPO = 0)
- And, when coupled with an active/active architecture:
  - Continuous availability (RTO ⟶ 0); this means no application outage across node failures!
  - Additionally, the elimination of *data collisions* when the application is active on all nodes

## *Coordinated Commits* Performs Well for Synchronous Replication:

- When the nodes are dispersed (e.g. for disaster tolerance); or
- When network loading/performance is important; or
- When the application I/O rates are high; or
- When the transactions are large (many I/O's); or
- When the number of nodes grows.

## The Key, Then, is to Implement Synchronous Replication for Reasonably "Well-Behaved" Applications

- For example, for an application that can scale as load increases (because to attain the same thruput, the application must scale)

# Business Continuity Case Studies (All Async)

## In Order of Increasing Availability (High to Continuous)

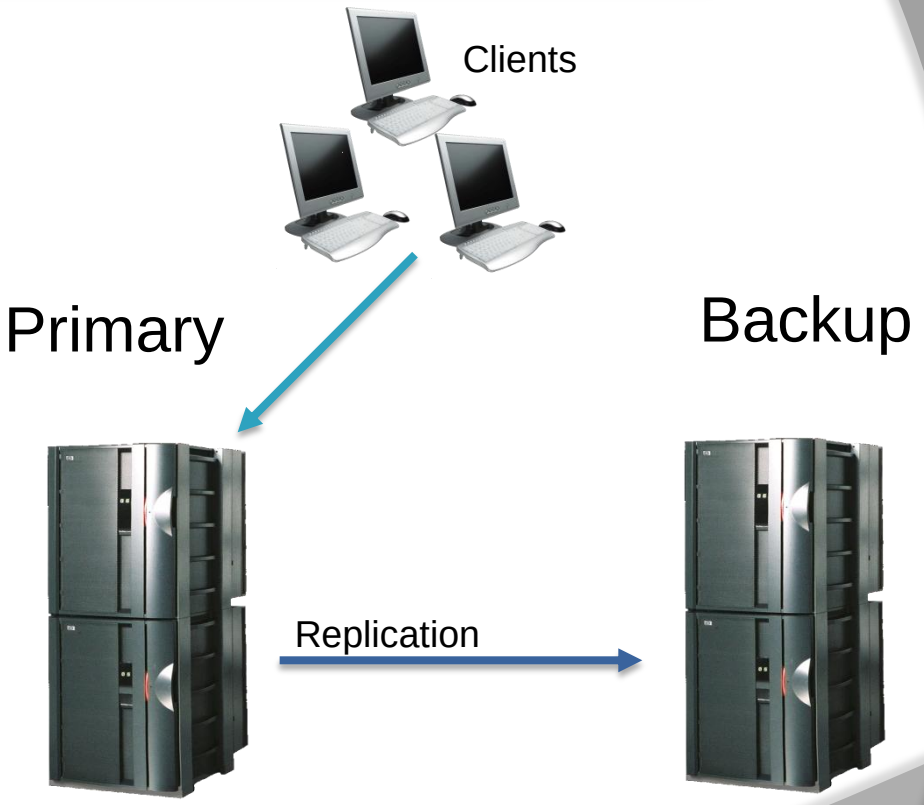
- Classic Disaster Recovery (Active/Passive)
- Sizzling Hot Takeover (Active/“Almost Active”)
- Continuous Availability (Active/Active)
  - Architectures That Avoid Data Collisions
    - Reciprocal Replication
    - Partitioned Application
    - Modify Database Keys
  - Architectures That Identify and Resolve Data Collisions
    - Data Content Resolution
    - Relative Replication
    - Designated Winner (Nodal Precedence)
- Miscellaneous Availability “Enhancers”
  - Asymmetric Capacity Expansion
  - Zero Downtime Migrations (ZDM)

# Business Continuity Case Study

## Classic Disaster Recovery

***Classic* DR:**

- Active → Passive
- Uni-Directional Replication
- Application *Active* on Primary Node Only
- Passive Node May be used for Reporting & Read-only OLQP

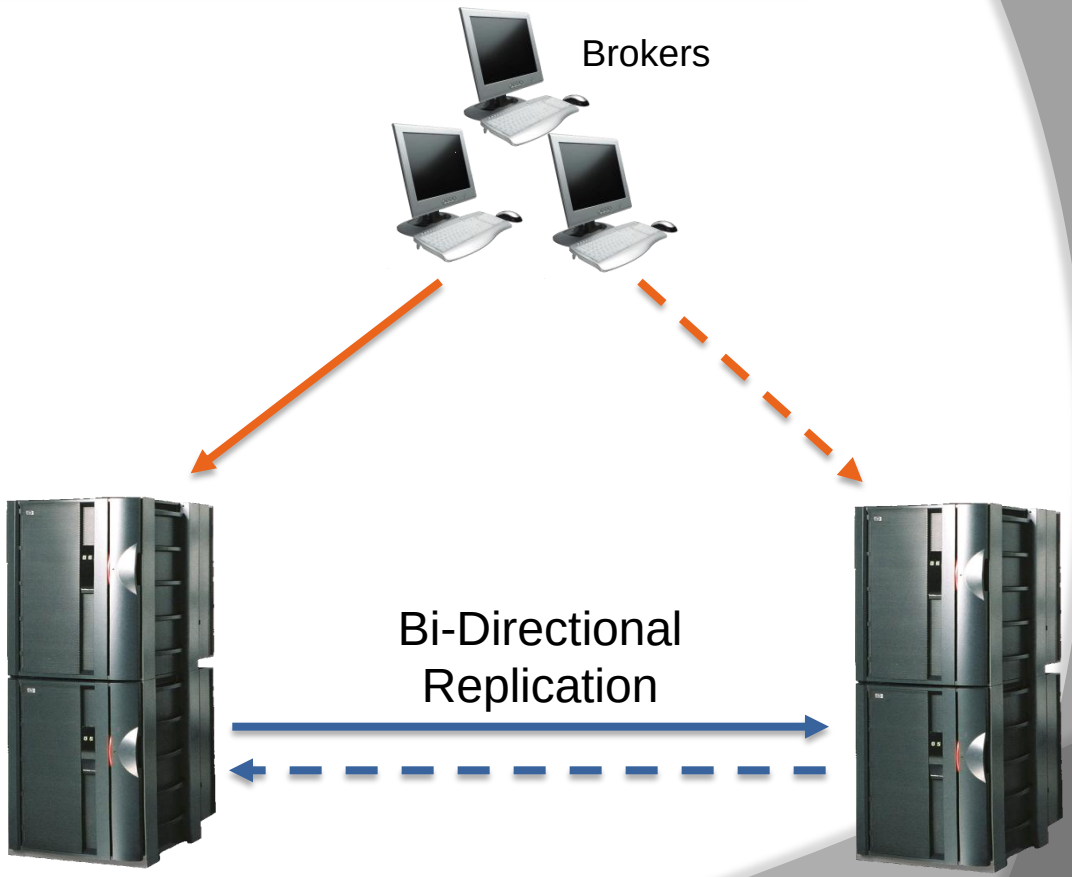


# Business Continuity Case Study

## Sizzling Hot Takeover

***Sizzling Hot Takeover*:**

- Active → "Almost Active"
- Target Appl Hot (Improves RTO)
- Bi-Dir Configured
- No Data Collisions
- Easy to Validate Backup (Submit Verification Tx's)
- Facilitates Failover Testing – Periodically "Just Do It"

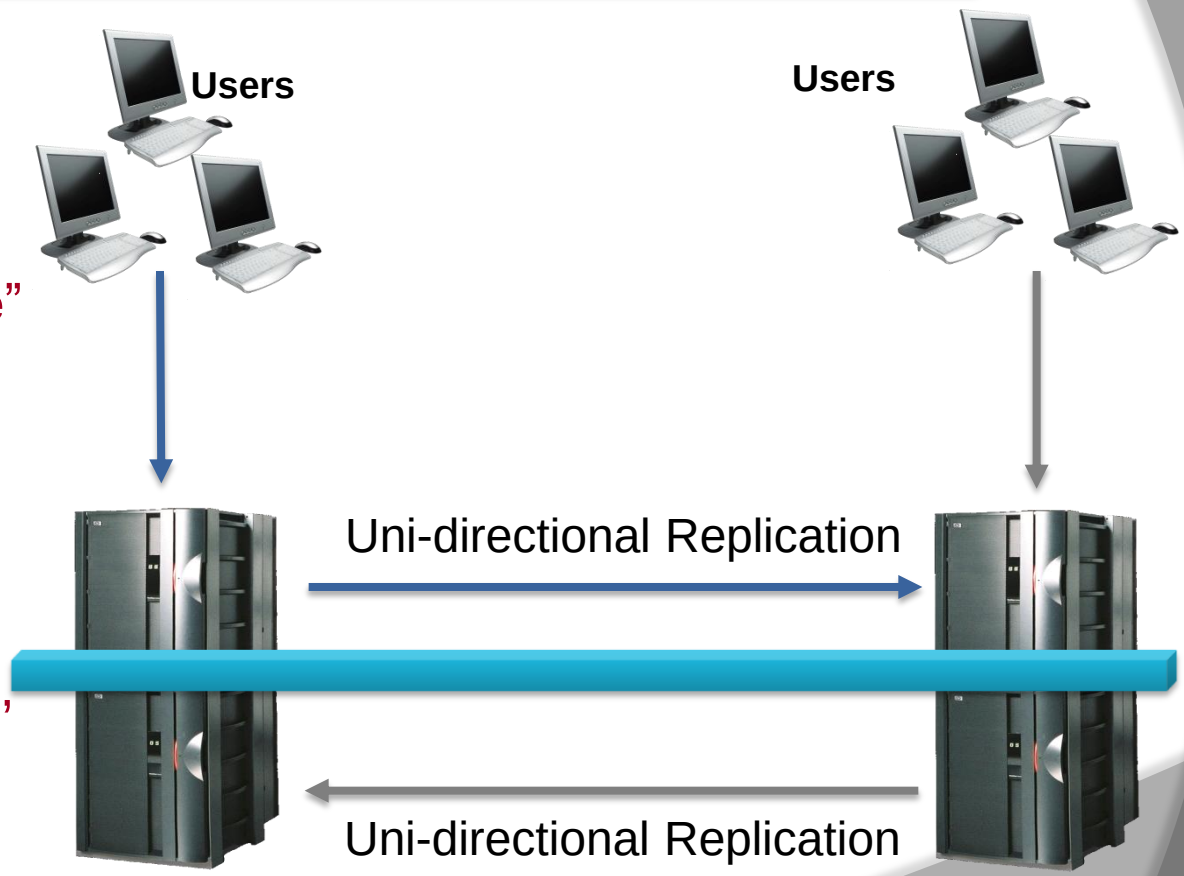


# Business Continuity Case Study

## All Nodes Active: Reciprocal Replication

***All Nodes Active:***

- **Not** "Active ↔ Active"
- 2 or more Uni-Directional DR Environments, Each Backing the Other(s) Up
- Separate Databases, Data Collisions Cannot Occur

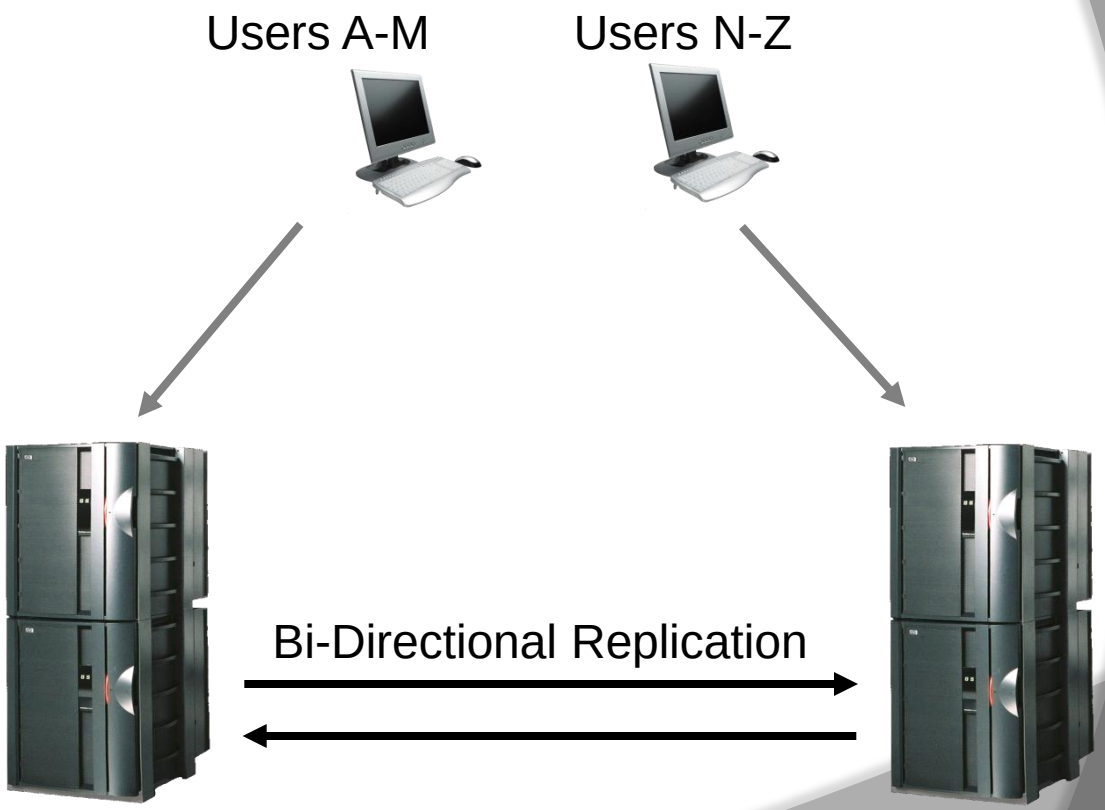


# Business Continuity Case Study

## Active/Active: Partitioned Application Users – No Data Collisions

***Partitioned Users:***

- Active ↔ Active
- All Applications Active
- Bi-Directional
- Users Load Balanced Across Nodes (pre-assigned a "primary" node)
- No Data Collisions

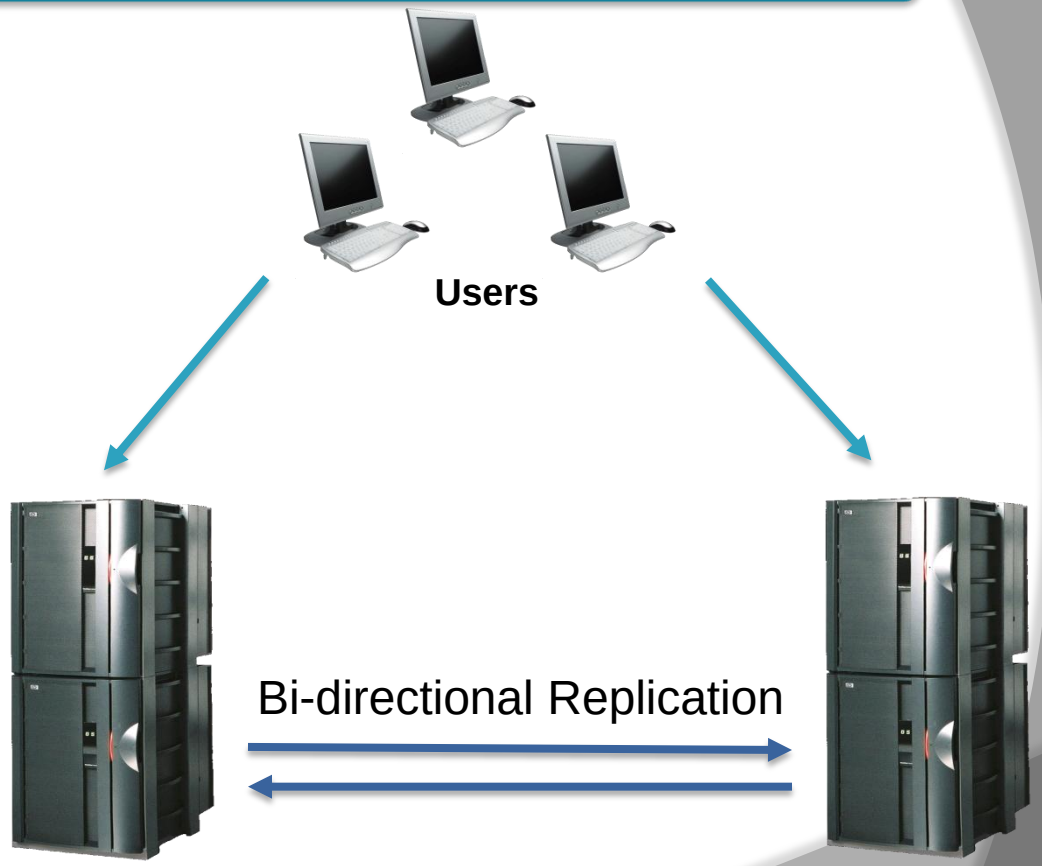


# Business Continuity Case Study

## Active/Active: Modify Application Database – No Data Collisions

***Modify Database:***

- Active ↔ Active
- All Applications Active
- Users Load Balanced Across Nodes
- Application Primarily INSERTs
- No Data Collisions as Node ID is added to the Database Keys

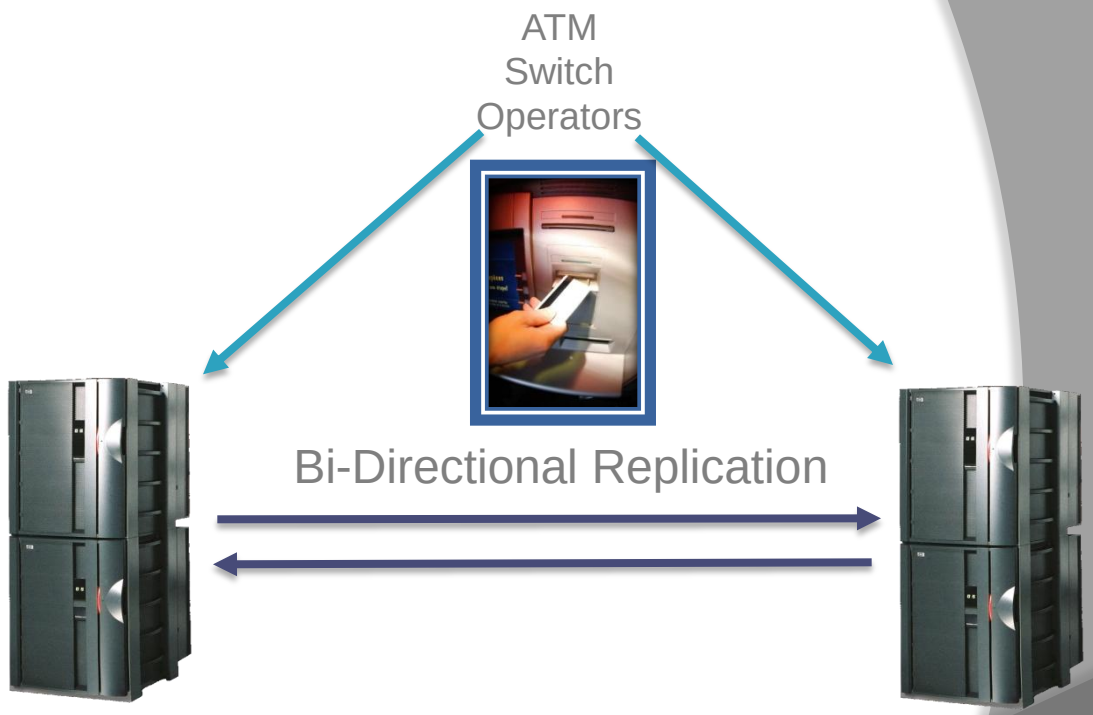


# Business Continuity Case Study

## Active/Active: Data Content Resolution

***Data Content:***

- Active ↔ Active
- All Applications Active
- Requests Load Balanced Across Nodes
- Data Collisions Occur, Resolved via Row Contents (e.g. Most Recent Update Timestamp)



# Business Continuity Case Study

## Active/Active: Relative Replication

***Relative:***

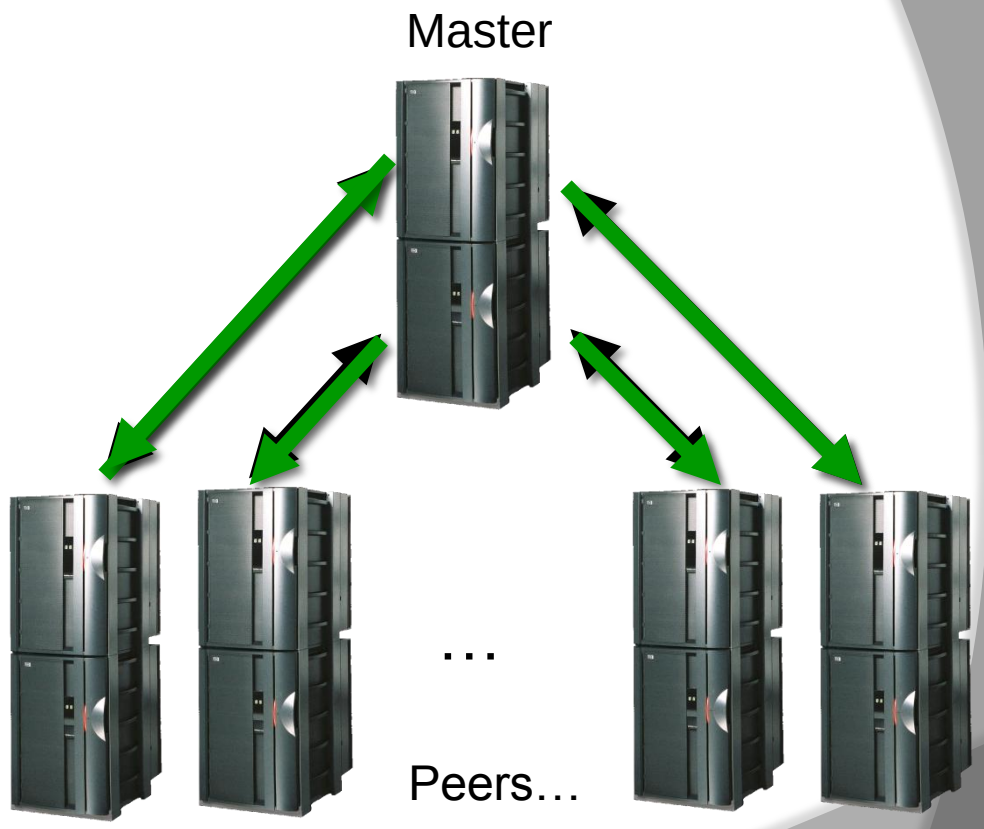
- Active ↔ Active
- All Applications Active
- Requests Load Balanced Across Nodes
- Data Collisions Occur, Resolved via Relative Replication (e.g. data deltas for numeric fields; text uses Absolute Replication)

# Business Continuity Case Study

## Active/Active: Designated Winner

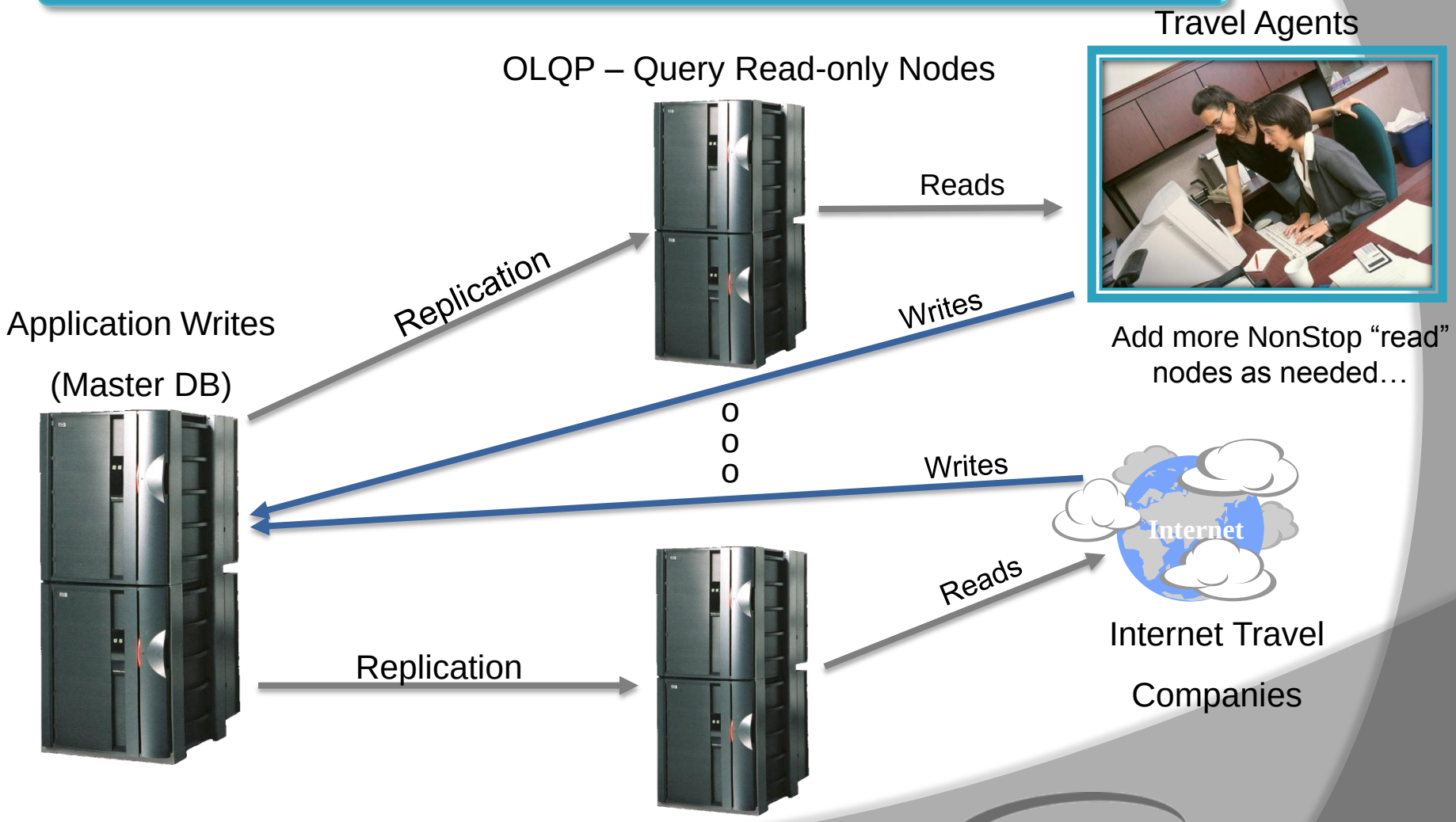
**Designated Winner:**

- Active ↔ Active
- All Applications Active
- Requests Load Balanced Across Nodes
- Data Collisions Occur, Resolved via “First to Update the Master Wins” Rule.



# Asymmetric Capacity Expansion

## Active/Active: Asymmetric Capacity Expansion



# For More Information

## Breaking the Availability Barrier Series

### Volume 1:

- Survivable Systems for Enterprise Computing

### Volume 2:

- Achieving Century Uptimes with Active/Active

### Volume 3:

- Active/Active Systems in Practice

### Volume 4:

- Continuous Availability for the 21st Century (Available 2010)

# For More Information

## Connect ***Connection*** Series & Website

### ITUG High Availability Series

- 6-Part Series on Availability Fundamentals (Starting Nov/Dec 2002)

### Connection Continuous Availability Series

- Ongoing Series (23 so far) on Advanced Availability Topics (Starting Nov/Dec 2006)

### Connection Availability Corner

- Periodic Articles on Pertinent Business Continuity Topics

### Gravic Website

- See White Papers in www.gravic.com/shadowbase/literature area

# Questions?


## Gravic, Inc.

**301 Lindenwood Drive**
**Suite 100**
**Malvern, PA 19355 USA**

**Shadowbase@gravic.com**
**SBSales@gravic.com**
**www.gravic.com**

**Phone: +1.610.647.6250**
**Fax: +1.610.647.7058**