Managing High Volume Replication Tiers
Edward Archibald
Chief Architect
Emic

MySQL Users Conference, 2005
April 20, 2005
Overview

• Why replicate?
• Replication Technology Landscape
• Challenging Issues
• Commercial Examples/Case Studies
• Questions and Answers
## Replication Technology Matrix

<table>
<thead>
<tr>
<th></th>
<th>Data replication/clustering</th>
<th>Synchronous replication</th>
<th>Asynchronous Replication</th>
<th>Eager replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>High availability</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Application Transparency</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Load balancing for queries</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>No data loss when Node fails</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Low latency</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Why replicate?
Why replicate?

• A definition: the maintenance of exact copies of specific sets of data
• A strategy for fault tolerance
• Enabling technology for scalability
• A solution for concurrency issues
Why replicate?

• Replication is THE core technology for building highly available, scalable database applications
Definitions

• Query – a read operation (select)
• Update – a write operation (insert, update/replace, delete)
• Transaction – a group of operations which exhibit ACID properties
Replication Technology
Landscape
Data Replication

• Disk-block based replication
• Hot-standby solution
• Works in disaster recovery scenarios
• Active-active applications not possible
Data Replication + Failover

- Utility clustering solutions leverage data replication to provide HA
- Provide single system image for Apps
- Automate planned and unplanned failover
Data Replication + Utility Clustering
Database Replication

- Three major categories:
  - Synchronous
  - Asynchronous (passive)
  - Eager (active)
Synchronous Replication

- Updates sent to all participants synchronously
- Locks acquired across all db instances
- Two-phase commit insures replica consistency
Synchronous Replication via X/Open DTP
Synchronous Replication

- Enabled by X/Open DTP compliance
- Contraindications*
  - May have high latency per request
  - Failure may leave resources locked
  - Not transparent to applications
Asynchronous Replication

• Primary backup model:
  – Updates are executed at a ’primary’ database server
  – Execution results in logging of changes
  – ’Log shipping’ propagates changes to ’backup’ replicas
Asynchronous Replication

- Little or no latency added to original update
- Simplified transaction serialization
- Can tolerate long network outages
- Handling NOW() etc. simplified
Asynchronous Replication

• Contraindications
  – Difficult to use for scaling some types of applications – ’stale’ reads
  – Replicas may get ’swamped’ during peak load or with high write ratios
  – Potential data loss if primary is lost
Eager Replication

• Updates are propagated to all nodes using enhanced atomic multicast
• All nodes receive all updates in the same order
• Deterministic execution of updates
Single System Image Cluster

Node A
- Database Server (Deterministic Execution)
- Interceptor
- Group Communication Service

Node B
- Database Server (Deterministic Execution)
- Interceptor
- Group Communication Service

Load Balancer Layer

Updates and Queries

Applications

Eager Replication

MySQL®

EMIC
Mission Critical Open Source
Eager Replication

• Per-request latency very low
• No data loss if any node is lost – update is executed everywhere or nowhere
• Very good read scalability with strict query consistency
Eager Replication

• Contraindications
  – Limited scalability for write intensive applications
  – Potential for concurrency-related performance issues
Challenging Issues
Challenging Issues

• Strict query consistency for non-synchronous replication techniques
  – Detect ’stale’ data and retry
  – Order queries so that they always follow potentially conflicting updates
Primary Backup Asynchronous Replication

MySQL

emic
Mission Critical Open Source
Eager Replication

MySQL

emc
Mission Critical Open Source
Challenging Issues

• Swamping of replicas
  – Implement flow control i.e. applications monitor backup replicas for currency, defer new requests
  – Implement flow control within the replicated database system
Primary Backup Asynchronous Replication

MySQL®

EMIC
Mission Critical Open Source
Commercial/Open Source Examples and Case Studies
Synchronous Replication: MySQL Cluster

- MySQL Cluster
  - Integrated with MySQL as NDB table type
  - Database fits in main memory
  - Optimized lock acquisition, 2PC
  - Write scalability via data fragmentation
Synchronous Replication: MySQL Cluster

• Bredbandsbolaget (B2) – Sweden’s largest broadband internet provider
• MySQL cluster provides user profile and authorization services for 175,000 users
Asynchronous Replication: MySQL Native Replication

- MySQL 'native' replication
  - Primary backup model
  - Very easy to set up and administer
  - High volumes with low latency
  - Multiple slaves per master
Asynchronous Replication: MySQL Native Replication

- CitySearch
  - Two data centers
  - Oracle DB replicated to MySQL masters
  - Each MySQL master has 10 slaves
  - Slaves serve 100 million queries/month
Eager Replication: Emic m/cluster

- Simple app migration via single system image
- GCS based active replication
- Strict consistency query load balancing
Emic m/cluster
ADSL Authentication

• Scarlett BV: ADSL service provider
• Fast, secure, initial subscriber login for 1.5 million subscribers
• Must handle peak loads
• Must scale for new users, new apps
EMIC EAC within Scarlet Access Authentication Platform

Customer → Modem → Internet → Network Access Systems
→ Authentication Service → RADIUS Protocol
Protocol Authentication Authorisation Accounting

Customer → ADSL → Network Access Systems

Customer → Wireless Router

MySQL Application

EMIC EAC Application

MySQL Database Servers

Subscriber and Accounting data
# Replication Technology Matrix

<table>
<thead>
<tr>
<th>Feature</th>
<th>Data replication/clustering</th>
<th>Synchronous replication</th>
<th>Asynchronous Replication</th>
<th>Eager replication</th>
</tr>
</thead>
<tbody>
<tr>
<td>High availability</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Application Transparency</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Load balancing for queries</td>
<td>No</td>
<td>In some cases</td>
<td>For some apps</td>
<td>Yes</td>
</tr>
<tr>
<td>No data loss when Node fails</td>
<td>Only with synchronous writes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Low latency</td>
<td>Yes</td>
<td>Not for XA</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Questions and Answers