Replication in Databases and Distributed Systems
Course: CS655

RABIET Louis

Colorado State University

Thursday 5 December 2013
1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
- Eager and lazy replication are not appropriate solutions [next slides].
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
- Eager and lazy replication are not appropriate solutions [next slides].
  - Eager: Synchronization then commit.
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
- Eager and lazy replication are not appropriate solutions [next slides].
  - Eager: Synchronization then commit.
  - Lazy: Asynchronous propagation.
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are supposed to have a copy of the data (no partition, no scaling up).
- Eager and lazy replication are not appropriate solutions [next slides].
  - Eager: Synchronization then commit.
  - Lazy: Asynchronous propagation.
- Fixed number of objects in the database.
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
- Eager and lazy replication are not appropriate solutions [next slides].
  - Eager: Synchronization then commit.
  - Lazy: Asynchronous propagation.
- Fixed number of objects in the database.
- Fixed number of nodes (but may be unavailable).
Analysis of replication in DB [GH96]

Hypothesis

- All the nodes are suppose to have a copy of the data (no partition, no scaling up)
- Eager and lazy replication are not appropriate solutions [next slides].
  - Eager: Synchronization then commit.
  - Lazy: Asynchronous propagation.
- Fixed number of objects in the database.
- Fixed number of nodes (but may be unavailable).
- No delay in communications.
Eager Replication

Just one node.

\[
\text{Node\_Deadlock\_Rate} \approx \text{Trans\_Deadlock\_Rate} \times \text{Nb\_Trans}
\]

\[
\frac{\text{Prob\_Deadlock}}{\text{Lifetime\_One\_Trans}}
\]
**Eager Replication**

*Just one node.*

\[
\text{Node}_\text{Deadlock}_\text{Rate} \approx \frac{\text{Prob}_\text{Deadlock}}{\text{Lifetime}_\text{OneTrans}} \times \text{Trans}_\text{Deadlock}_\text{Rate} \times \text{Nb}_\text{Trans}
\]

\[
\text{Trans}_\text{Per}_\text{sec} \times \text{Nb}_\text{Actions} \times \text{OneAction}_\text{Time}
\]
Eager Replication

Just one node.

\[
\text{Node Deadlock Rate} \approx \text{Trans Deadlock Rate} \times \text{Nb Trans}
\]

\[
\frac{\text{Prob Deadlock}}{\text{Lifetime One Trans}}
\]

\[
\text{Trans Per sec} \times \text{Nb Actions} \times \text{One Action Time}
\]
Eager Replication

*Just one node.*

\[
\text{Node\_Deadlock\_Rate} \approx \frac{\text{Trans\_Deadlock\_Rate} \times \text{Nb\_Trans}}{	ext{Prob\_Deadlock} \times \text{Lifetime\_One\_Trans}}
\]

\[
\text{Prob\_Deadlock} \approx (\text{Prob\_Waiting})^2 \times \text{Nb\_Trans}
\]

\[
\text{Prob\_Deadlock} = \frac{\text{Trans\_Per\_sec} \times \text{Nb\_Actions} \times \text{OneAction\_Time}}{	ext{Lifetime\_One\_Trans}}
\]
Eager Replication

Generalization.

\[ \text{Nb\_Trans'} = \text{Trans\_Per\_sec'} \times \text{Lifetime\_OneTrans'} \]

- \[ \text{Trans\_Per\_sec} \times \text{Nb\_Nodes} \]
Eager Replication

Generalization.

\[
\text{Nb}_\text{Trans}' = \text{Trans}_\text{Per}_\text{sec}' \times \text{Lifetime}_\text{OneTrans}'
\]

- \(\text{Trans}_\text{Per}_\text{sec} \times \text{Nb}_\text{Nodes}\)
- \(\text{Nb}_\text{Actions} \times \text{OneAction}_\text{Time} \times \text{Nb}_\text{Nodes}\)
Eager Replication

**Generalization.**

\[ \text{Nb}_\text{Trans}' = \text{Trans}_\text{Per}_\text{sec}' \times \text{Lifetime}_\text{OneTrans}' \]

- \( \text{Trans}_\text{Per}_\text{sec} \times \text{Nb}_\text{Nodes} \)
- \( \text{Nb}_\text{Actions} \times \text{OneAction}_\text{Time} \times \text{Nb}_\text{Nodes} \)
Eager Replication

Generalization.

\[ \text{Nb}_\text{Trans'} = \text{Trans}_\text{Per}_\text{sec'} \times \text{Lifetime}_\text{OneTrans'} \]

- \( \text{Trans}_\text{Per}_\text{sec} \times \text{Nb}_\text{Nodes} \)
- \( \text{Nb}_\text{Actions} \times \text{OneAction}_\text{Time} \times \text{Nb}_\text{Nodes} \)

Scalability issue

\( \text{Nb}_\text{Trans'} \) in \( \text{Nb}_\text{Nodes}^2 \) order
Eager Replication

Generalization.

\[ \text{Eager Deadlock} = \frac{\text{Prob Deadlock Eager}}{\text{Lifetime OneTrans'}} \times \text{Nb Trans'} \]
Eager Replication

Generalization.

$$\text{Eager Deadlock} = \frac{\text{Prob Deadlock Eager}}{\text{Lifetime OneTrans'}} \times \text{Nb_Trans'}$$

$$\text{Prob Deadlock Eager} \approx (\text{Prob Waiting})^2 \times \text{Nb_Trans'}$$
Eager Replication

Generalization.

\[
\text{Eager Deadlock} = \frac{\text{Prob Deadlock Eager}}{\text{Lifetime OneTrans'}} \times \text{Nb_Trans'}
\]

\[
\text{Prob Deadlock Eager} \approx (\text{Prob Waiting})^2 \times \text{Nb_Trans'}
\]

Scalability issue
Eager Deadlock in Nb Nodes \(^3\) order
Lazy Group Replication

Motivation

- scalable.
Lazy Group Replication

Motivation

- scalable.
- support disconnection of nodes.
Lazy Group Replication

Motivation

- scalable.
- support disconnection of nodes.
Lazy Group Replication

Motivation

- scalable.
- support disconnection of nodes.

Reminder

- Any node can update data (no master) = Group Replication.
Lazy Group Replication

Motivation

- scalable.
- support disconnection of nodes.

Reminder

- Any node can update data (no master) $\Rightarrow$ Group Replication.
- Reconciliation of conflict (timestamps are used) is needed (no deadlock in lazy group schema).
Lazy Group Replication

\[
\text{Lazy\_Reconciliation\_Rate} = \frac{\text{Prob\_Collision} \times \frac{\text{Inbound\_Updates} \times \text{Outbound\_Updates}}{\text{DB\_Size}}}{\text{Disconnection\_Time}} \times \frac{\text{Nb\_Nodes}}{\text{Disconnection\_Time}}
\]

(in \( o(\text{Nb\_Nodes}) \))
Lazy Group Replication

\[
\text{Lazy_Reconciliation_Rate} = \frac{\text{Prob_Collision} \times \frac{\text{Inbound_Updates} \times \text{Outbound_Updates}}{\text{DB Size}}}{\text{Nb_Nodes} / \text{Disconnection_Time}}
\]

(in $o(\text{Nb_Nodes})$)
<table>
<thead>
<tr>
<th>Lazy Master</th>
</tr>
</thead>
</table>

- A master is assigned to each object.
Lazy Master

- A master is assigned to each object.
- Conflict are in $o(\text{Nb\_Nodes}^2)$. 
Lazy Master

- A master is assigned to each object.
- Conflict are in $o(Nb\_Nodes^2)$.
- No support for disconnection.
Two-Tier Replication

- Slight modification of the lazy master replication schema.
Two-Tier Replication

- Slight modification of the lazy master replication schema.
- Restriction: at most a mobile node is involved in the transaction.
Two-Tier Replication

- Slight modification of the lazy master replication schema.
- Restriction: at most a mobile node is involved in the transaction.
- A mobile (in the case where it is the master) while connect to a base node (that contains a replica of the data) and will update his content with this new data.
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
Motivation

- Databases and distributed systems use replication.
Motivation

- Databases and distributed systems use replication.
- One for Performance. One for failure resilience.
Motivation

- Databases and distributed systems use replication.
- One for Performance. One for failure resilience.
- Idea is to compare them.
5 Phases in replication

- Request.
5 Phases in replication

- Request.
- Server Coordination.
5 Phases in replication

- Request.
- Server Coordination.
- Execution.
5 Phases in replication

- Request.
- Server Coordination.
- Execution.
- Agreement Coordination.
5 Phases in replication

- Request.
- Server Coordination.
- Execution.
- Agreement Coordination.
- Response.
Distributed System replication

- Active replication.
Distributed System replication

- Active replication.
- Passive replication.
Distributed System replication

- Active replication.
- Passive replication.
- Semi-active replication.
Database replication *One operation.*

- **Eager primary copy.**
Database replication *One operation.*

- Eager primary copy.
- Eager update everywhere with distributed locking.
Database replication *One operation.*

- Eager primary copy.
- Eager update everywhere with distributed locking.
- Eager update everywhere based on atomic broadcast.
Database replication *One operation.*

- Eager primary copy.
- Eager update everywhere with distributed locking.
- Eager update everywhere based on atomic broadcast.
- Lazy update everywhere.
Database replication *Several operations.*

- Eager primary copy.
Database replication *Several operations.*

- **Eager primary copy.**
- **Eager update everywhere with distributed locking.**
Database replication *Several operations.*

- Eager primary copy.
- Eager update everywhere with distributed locking.
- Certification replication.
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
Application [hHBR⁺04]

- Interested in recovery.
Application [hHBR+04]

- Interested in recovery.
- Precise > Rollback > Gap.
Application \textsuperscript{[hHBR\textsuperscript{+}04]}

- Interested in recovery.
- Precise \(>\) Rollback \(>\) Gap.
- Streams of data.
Application [hHBR$^+04$]

- Interested in recovery.
- Precise $>$ Rollback $>$ Gap.
- Streams of data.
- Pairs of primary/secondary.

![Diagram of server pairs and streams](image-url)
Gap Recovery

- In case of failure or partial communication, just ignore the queries.
Gap Recovery

- In case of failure or partial communication, just ignore the queries.
- Called amnesia.
Gap Recovery

- In case of failure or partial communication, just ignore the queries.
- Called amnesia.
- No recovery.
Passive standby

- Sending the input queues, operators and output queues to the secondary node.
Passive standby

- Sending the input queues, operators and output queues to the secondary node.
- To avoid suspension during check-pointing: sweep line.
Passive standby

- Sending the input queues, operators and output queues to the secondary node.
- To avoid suspension during check-pointing: sweep line.
- Strongest property for recovery.
Upstream backup

- Upstream nodes act as backup for downstream nodes.
Upstream backup

- Upstream nodes act as backup for downstream nodes.
Upstream backup

- Upstream nodes act as backup for downstream nodes.
Active standby

- Same as passive standby except that upstream secondary nodes receive the tuples and are processing it.
Active standby

- Same as passive standby except that upstream secondary nodes receive the tuples and are processing it.
- Because of non deterministic operators, input must be recorded to keep the relation input, output.
Some extensions to make precise recovery

- Passive standby: verification with downstream nodes.
Some extensions to make precise recovery

- Passive standby: verification with downstream nodes.
- Active standby: distribute information to replay operations.
Some extensions to make precise recovery

- Passive standby: verification with downstream nodes.
- Active standby: distribute information to replay operations.
- Upstream Backup: using the watermarks.
## Experimentation

<table>
<thead>
<tr>
<th></th>
<th>recovery_time</th>
<th>bandwidth_overhead</th>
<th>processing_overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amnesia</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Passive</td>
<td>$K + Qp$</td>
<td>$f_1(1/M, C)$</td>
<td>$f_2(1/M, C)$</td>
</tr>
<tr>
<td>Upstream</td>
<td>$K + Q'p$</td>
<td>$f_3(1/M, c)$</td>
<td>$f_4(1/M, \text{ops, paths})$</td>
</tr>
<tr>
<td>Active</td>
<td>$\epsilon$</td>
<td>$100% + f_3(1/M, c)$</td>
<td>$100% + 2 \times f_4(1/M, \text{ops, paths})$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Network Type</th>
<th>Result</th>
<th>Upstream Backup</th>
<th>Active Standby</th>
<th>Passive Standby</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeatabile</td>
<td>Bw overhead (%)</td>
<td>0.64</td>
<td>100.96</td>
<td>101.27</td>
</tr>
<tr>
<td></td>
<td>Rec. time (ms)</td>
<td>47.62</td>
<td>1.80</td>
<td>45.88</td>
</tr>
<tr>
<td>Convergent-capable</td>
<td>Bw overhead</td>
<td>0.64</td>
<td>100.96</td>
<td><strong>111.55</strong></td>
</tr>
<tr>
<td></td>
<td>Rec. time</td>
<td><strong>69.86</strong></td>
<td>0.07</td>
<td>48.88</td>
</tr>
<tr>
<td>Non-deterministic</td>
<td>Bw overhead</td>
<td>1.28</td>
<td>101.91</td>
<td>101.90</td>
</tr>
<tr>
<td></td>
<td>Rec. time</td>
<td>50.92</td>
<td>1.82</td>
<td>47.24</td>
</tr>
</tbody>
</table>
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
CRDT

Commutativity $\implies$ Convergence simpler
CRDT

- SEC (Strong eventual Consistency).
CRDT

- SEC (Strong eventual Consistency).
  - Eventual delivery $\forall i, j, f \in c_i \implies \diamond f \in c_j$
CRDT

- SEC (Strong eventual Consistency).
  - Eventual delivery $\forall i, j, f \in c_i \implies \Diamond f \in c_j$
  - Strong convergence $\forall i, j, c_i \in c_j \implies s_i \equiv s_j$
CRDT

- SEC (Strong eventual Consistency).
  - Eventual delivery $\forall i, j, f \in c_i \implies \diamond f \in c_j$
  - Strong convergence $\forall i, j, c_i \in c_j \implies s_i \equiv s_j$
  - Termination
CRDT

- SEC (Strong eventual Consistency).
  - Eventual delivery $\forall i, j, f \in c_i \implies \diamond f \in c_j$
  - Strong convergence $\forall i, j, c_i \in c_j \implies s_i \equiv s_j$
  - Termination

- Without any conflict resolution.
CRDT

- SEC (Strong eventual Consistency).
  - Eventual delivery $\forall i, j, f \in c_i \implies \diamond f \in c_j$
  - Strong convergence $\forall i, j, c_i \in c_j \implies s_i \equiv s_j$
  - Termination

- Without any conflict resolution.

- $n - 1$ failures possible.
Examples of CDRT

Vectors counters.

\((\mathbb{N}^n, [0, \ldots, 0], \geq^n, value, inc, max^n)\)

if \(p_i\) can only update the i-th element in the vector \(\equiv\) vector-clock
Sufficient condition

\[
\begin{align*}
\{ & \text{Eventual delivery} \\
& \text{Termination} \\
& \text{Monotonic semi-lattice} \}
\Rightarrow \text{SEC}
\end{align*}
\]
Treedoc [PMSL09]
Treedoc [PMSL09]
Treedoc [PMSL09]
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
Hybrid Quorum [CML$^+$06]

- Use a quick quorum if no contention.
Hybrid Quorum [CML+06]

- Use a quick quorum if no contention.
- BFT [CL02] if contention.
Hybrid Quorum [CML\textsuperscript{+}06]

- Use a quick quorum if no contention.
- BFT [CL02] if contention.
- Combine quorum and agreement based state-machine replication.
Normal Case

- Write operations = 2 phases.
Normal Case

- Write operations = 2 phases.
- Read operations = 1 phase
Normal Case

- Write operations = 2 phases.
- Read operations = 1 phase
- $3f + 1$ replica.
Normal Case

- Write operations = 2 phases.
- Read operations = 1 phase
- $3f + 1$ replica.
- Phase Write1 = timestamp $t$ obtained using a quorum.
Normal Case

- Write operations = 2 phases.
- Read operations = 1 phase
- $3f + 1$ replica.
- Phase Write1 = timestamp $t$ obtained using a quorum.
- Phase Write2 = Execution ($2f + 1$ must agree to execute the operation at $t$).
Contestation

- if Write2 has been completed, it is still assigned.
Contention

- if Write2 has been completed, it is still assigned.
- if Write1 has obtained $t$ then the operation will be executed at $t' \geq t$
Plan

1. Eager and lazy replications
2. Databases and distributed systems
3. Streaming application
4. CRDT
   - Treedoc: Cooperative editing
5. Hybrid Replication
6. Thoughts
7. CCL
CRDT

✓ Applicable only on commutative operations.
CRDT

- Applicable only on commutative operations.
- Not many use cases?
Streaming

✓ Number of replica = 2.
Streaming

✓ Number of replica = 2.

- Checkpointing.
Streaming

☑ Number of replica = 2.

- Checkpointing.
- Load Balancing between the two nodes.
Streaming

✓ Number of replica = 2.
- Checkpointing.
- Load Balancing between the two nodes.
- Several loops (image processing).
Streaming

- Number of replica = 2.
- Checkpointing.
- Load Balancing between the two nodes.
- Several loops (image processing).
- Path exploration.
<table>
<thead>
<tr>
<th>Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Eager and lazy replications</td>
</tr>
<tr>
<td>2. Databases and distributed systems</td>
</tr>
<tr>
<td>3. Streaming application</td>
</tr>
<tr>
<td>4. CRDT</td>
</tr>
<tr>
<td>- Treedoc: Cooperative editing</td>
</tr>
<tr>
<td>5. Hybrid Replication</td>
</tr>
<tr>
<td>6. Thoughts</td>
</tr>
<tr>
<td>7. CCL</td>
</tr>
</tbody>
</table>
Conclusion

Thank You for your attention!


