Outline

- Introduction
- Background
- Distributed DBMS Architecture
- Distributed Database Design
- Distributed Query Processing
- Distributed Transaction Management
  - Transaction Concepts and Models
  - Distributed Concurrency Control
  - Distributed Reliability
- Building Distributed Database Systems (RAID)
- Mobile Database Systems
- Privacy, Trust, and Authentication
- Peer to Peer Systems
Useful References


- B. Bhargava, *Concurrency Control in Database Systems*, IEEE Trans on Knowledge and Data Engineering, 11(1), Jan.-Feb. 1999
Optimistic Concurrency Control Algorithms

- Transaction execution model: divide into subtransactions each of which execute at a site
  - $T_{ij}$: transaction $T_i$ that executes at site $j$
- Transactions run independently at each site until they reach the end of their read phases
- All subtransactions are assigned a timestamp at the end of their read phase
- Validation test performed during validation phase. If one fails, all rejected.
Optimistic Concurrency Control Processing

Start → Read, Compute, And Write Local → Integrity Control & Local Validation → Success → Semi-Commit On Initiating Site → Success → Integrity Control & Global Validation → Success → Commit, Global Write Finish → Fail

Start → Read, Compute, And Write Local → Integrity Control & Local Validation → Fail

Start → Read, Compute, And Write Local → Integrity Control & Local Validation → Fail → Semi-Commit On Initiating Site → Fail
Transaction Types on a Site

Committed Transactions

Semi-Committed Transactions

Transactions Still Reading/Computing

Validating Transactions
Example of Locking vs Optimistic

\[
S(R_i) \cap S(W_j) \neq \emptyset \quad \text{AND} \\
\Pi(R_i) < \Pi(W_j) \\
\Rightarrow T_i \rightarrow T_j
\]

**Locking**
- \(R_i \ R_j \ W_i \ W_j\)

**Optimistic**
- \(R_i \ R_j \ W_i \ W_j\)
- \(R_i \ R_j \ W_j \ W_i\)
Example: \( h = R_1 R_2 W_2 R_3 W_3 \ldots R_n W_n W_1 \)

Locking: This history not allowed

\( W_2 \) is blocked by \( R_1 \)
\( T_2 \) cannot finish before \( T_1 \)

What if \( T_1 \) is a log trans. and \( T_2 \) is a small trans.?

\( T_1 \) blocks \( T_2 \); can block \( T_3 \) … \( T_n \) if \( (R_2 \cap W_3 \neq \emptyset) \)

Optimistic [Kung]

\( T_i \) (i = 2,…,n) commit. \( W_i \) saved for valid\(_n\)
\( R_1 \) validated with \( W_i \), \( T_1 \) aborted

\( h = R_1 R_2 W_2 \ldots R_n W_n W_1 \)
Optimistic Validation (first modification)

\[ h = R_1 R_2 W_2 R_3 W_3 \ldots R_n W_n W_1 \]

Try this or this switch

\( T_i \)'s can commit, \( W_i \) and \( R_i \) saved from validation
\( W_1 \) validates with \( W_i \) and \( R_i \)
\( T_1 \) aborted if validation fails (second modification)

\[ h = R_1 R_2 W_2 R_3 W_3 \ldots R_n W_n W_1 \]

Switch \( R_1 \) to the right after \( W_2, W_3, \ldots W_n \)
Switch \( W_1 \) to the left before \( T_n, T_{n-1}, \ldots T_2 \) (associated \( R_n \) and \( W_n \) etc.)

If \( R_1 \) and \( W_1 \) are adjacent, \( T_1 \) is successful

\[ h \equiv R_1 R_2 W_2 \ldots R_k W_k \ldots R_n W_n W_1 \]

\[ \equiv R_2 W_2 \ldots R_1 W_1 R_k W_k \ldots R_n W_n \]
Probability that two transactions do not share an object

\[
= \left( \frac{M - B_s}{M} \right) \times \left( \frac{M - B_s - 1}{M - 1} \right) \times \left( \frac{M - 2B_s + 1}{M - B_s + 1} \right)
\]

Lower bound on this problem

\[
= \left( \frac{M - 2B_s + 1}{M - B_s + 1} \right)^{B_s}
\]

Maximum problem that two transactions will share an object

\[
= 1 - \left( \frac{M - 2B_s + 1}{M - B_s + 1} \right)^{B_s}
\]

<table>
<thead>
<tr>
<th>BS</th>
<th>M</th>
<th>Probability of conflict</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>100</td>
<td>.0576</td>
</tr>
<tr>
<td>10</td>
<td>500</td>
<td>.0025</td>
</tr>
<tr>
<td>20</td>
<td>1000</td>
<td>.113</td>
</tr>
</tbody>
</table>

Probability of cycle

\[
= 0(\text{PC}^2)
\]

\approx \text{small}
Concurrency/Multiprogramming level is low

Example:

\[
\begin{align*}
I/O &= 0.005 \text{ seconds} \\
CPU &= 0.0001 \text{ seconds} \\
\text{Trans size} &= 5 \\
\text{Time to execute trans.} &= 0.0255 \text{ seconds}
\end{align*}
\]

For another trans. to meet this trans. in the system

\[
\text{Arrival rate} > \frac{1}{0.0255} \quad \text{or} \quad > 40 \text{ per second}
\]
Optimistic CC Validation Test

If all transactions $T_k$ where $ts(T_k) < ts(T_{ij})$ have completed their write phase before $T_{ij}$ has started its read phase, then validation succeeds

- Transaction executions in serial order

\[ T_k | R | V | W | \]
\[ T_{ij} | R | V | W | \]
Optimistic CC Validation Test

If there is any transaction $T_k$ such that $ts(T_k) < ts(T_{ij})$ and which completes its write phase while $T_{ij}$ is in its read phase, then validation succeeds if

$WS(T_k) \cap RS(T_{ij}) = \emptyset$

- Read and write phases overlap, but $T_{ij}$ does not read data items written by $T_k$

\[ T_k \quad \underline{R} \quad \underline{V} \quad \underline{W} \quad T_{ij} \quad \underline{R} \quad \underline{V} \quad \underline{W} \]
Optimistic CC Validation Test

If there is any transaction $T_k$ such that $ts(T_k) < ts(T_{ij})$ and which completes its read phase before $T_{ij}$ completes its read phase, then validation succeeds if $WS(T_k) \cap RS(T_{ij}) = \emptyset$ and $WS(T_k) \cap WS(T_{ij}) = \emptyset$.

- They overlap, but don't access any common data items.