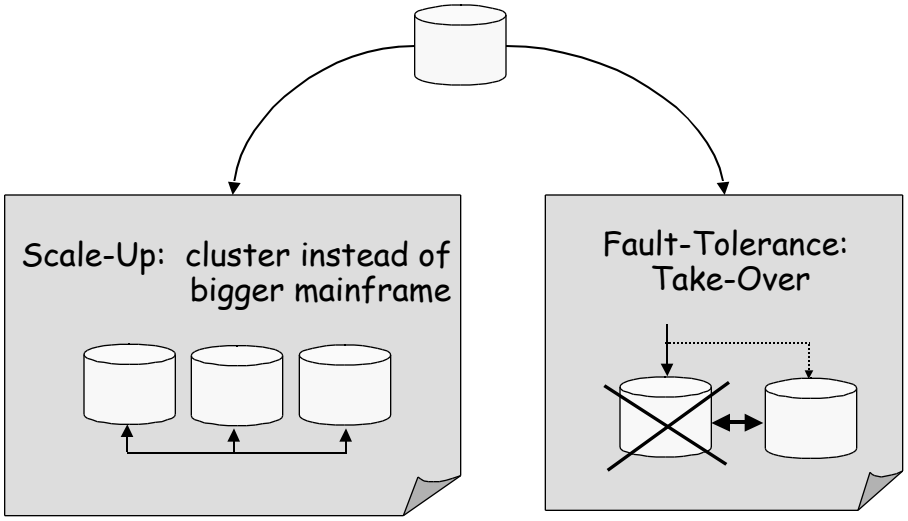
Consistent and Efficient Database Replication based on Group Communication

Bettina Kemme
School of Computer Science
McGill University, Montreal

Outline

- State of the Art in Database Replication
- Key Points of our Approach
- Overview of the Algorithms
- Implementation Issues
- Performance Results
- Ongoing Work

Replication -- Why?



Replica Control: Research and Reality

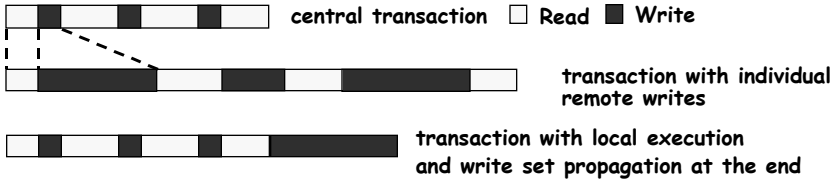
UPDATE WHEN \ UPDATE WHERE	Primary Copy	Update Everywhere
Eager		Globally correct Too expensive Deadlocks Quorum/ROWA Oracle Synchr. Repl
Lazy	Placement Strategies Feasible Inconsistent reads Configuration Restrictions Placement Strat. Sybase/IBM/Oracle	Feasible in some applications Inconsistent reads Reconciliation Weak Consistency Sybase/IBM/Oracle

Requirements

- Develop and apply appropriate techniques in order to avoid current limitations of eager update everywhere approaches
  - ☆ Keep flexibility of update everywhere
    - no restrictions on type of transactions and where to execute them
  - ☆ Consistency and fault-tolerance of eager replication
  - ☆ Good Performance
    - response time + throughput
- Straightforward, Implementable Solution
  - ☆ Easy integration in existing systems

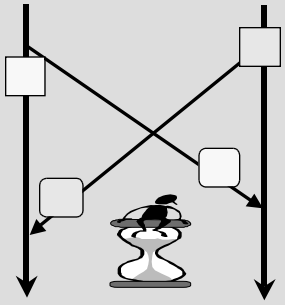
Response Time and Message Overhead

- Goal: Reduce number of messages per transaction
  - ☆ reduce response time
  - ☆ reduce message overhead
- Solution
  - ☆ local execution of transaction
  - ☆ bundle writes and send them in a single message at the end of the transaction (as done in lazy schemes)



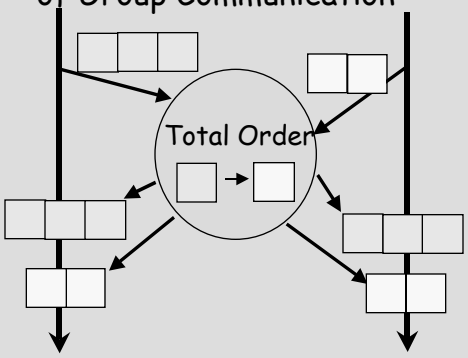
Ordering Transactions

- Before: uncoordinated message delivery; danger of deadlocks



- Before: 2-phase-commit

- Now: pre-order txns by using total order multicast of Group Communication

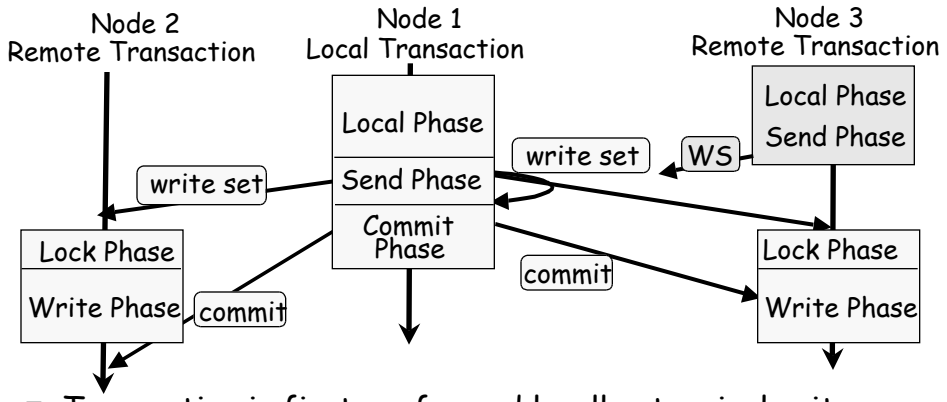


- Now: Independent execution at the different sites

Group Communication Systems

- Group Communication:
  - ☆ Multicast
  - ☆ Delivery order (FIFO, causal, total, etc.)
  - ☆ Reliable delivery: on all nodes vs. on all available nodes
  - ☆ Membership control
  - ☆ ISIS, Totem, Transis, Phoenix, Horus, Ensemble, ...
- Goal: Exploit rich semantics of group communication on a low level of the software hierarchy

Replica Control based on 2Phase Locking



- Transaction is first performed locally at a single site.
- Writes are sent in one message to all sites at the end of transaction.
- Write messages are totally ordered.
- Serialization order obeys total order.

Concurrency Control

- One possible Solution: Given a transaction T
  - ☆ Local Phase: T acquires standard local read and write locks
  - ☆ Send Phase: Send write set using total order multicast
  - ☆ Upon reception of write set of T on local node
    - Commit Phase: multicast commit message
  - ☆ Upon reception of write set of T on remote node
    - Lock Phase: request all write locks in a single step; if there is a local transaction T' with conflicting lock and T' is still in local phase or send phase, abort T'. If T' in send phase, multicast abort
    - Write Phase: apply updates of T
  - ☆ Upon reception of commit/abort message of T on remote node, terminate T accordingly
- For two transactions of same node: 2 phase locking.
- For concurrent transactions of different nodes: optimistic scheme with early conflict detection: when write set of one transaction is delivered the conflict is detected.
- Adjustment to other concurrency control schemes possible

Message Delivery Guarantees

- Uniform-Reliable Delivery
  - ☆ If a site delivers a message, all non-faulty sites deliver the message
  - ☆ Correctness on all sites (faulty or non-faulty): when a transaction commits at any site then it commits at all non-faulty sites
  - ☆ High message delay
- Reliable Delivery
  - ☆ If a non-faulty site (non-faulty for a sufficiently long time) delivers a message it is delivered at all non-faulty sites.
  - ☆ Correctness in the failure-free case
  - ☆ In case of failures:
    - all non-faulty sites commit the same set of transactions
    - a transaction might be committed at a faulty site (shortly before failure) and it is not committed at the other sites.
  - ☆ Low message delay

A Suite of Replication Protocols

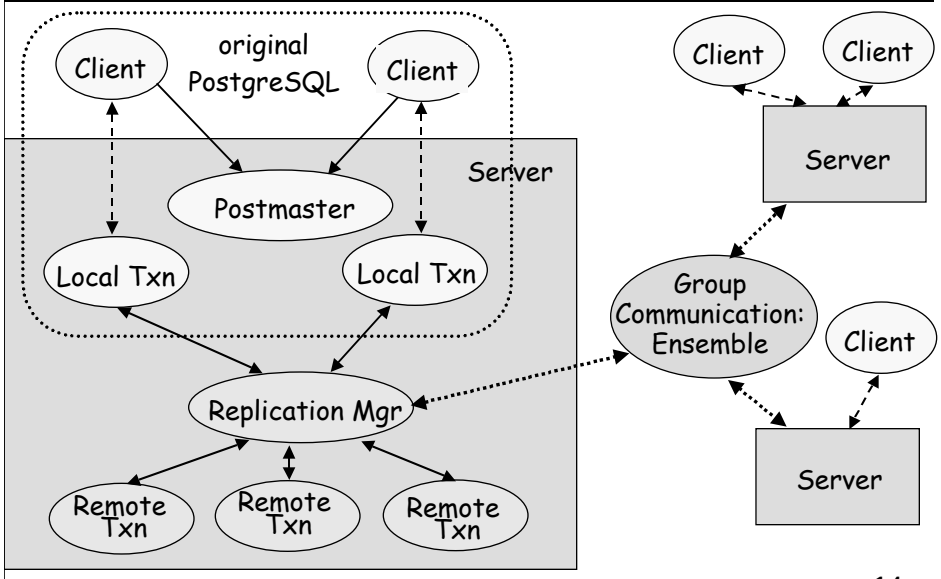
	Uniform Reliable	Reliable
Serializability	**SER-UR**	**SER - R**
Cursor Stability	**CS-UR**	**CS - R**
Snapshot Isolation	**SI-UR**	**SI - R**
Hybrid	**HYB-UR**	**HYB - R**

- The solutions provide
  - ☆flexibility
  - ☆accepted correctness criteria

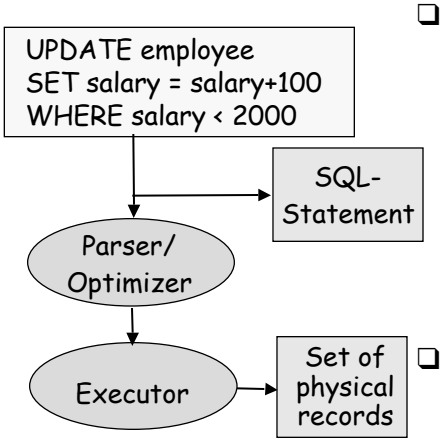
Implementation

- Integration of our replica control approach into the database system PostgreSQL
- Purpose - We wanted to answer the following questions
  - ☆ Can the abstract protocols really be mapped to concrete algorithms in a relational database?
  - ☆ How difficult is it to integrate the replication tool into a real database system?
  - ☆ What is the performance in a real cluster environment?

Architecture of Postgres-R

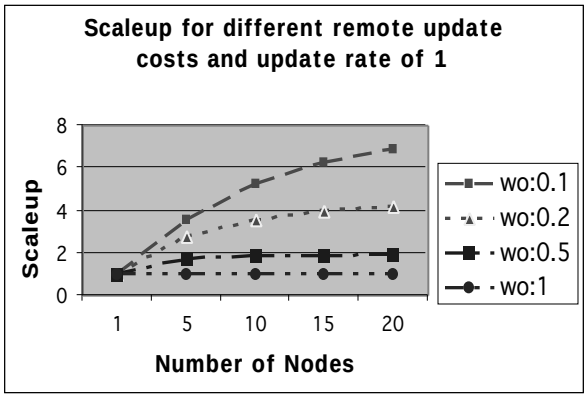


Write Set Messages



- Send SQL statements and reexecute at all sites
  - ☆ Simple
  - ☆ Small message size
  - ☆ High execution overhead on all site
  - ☆ Problem with locks for implicit reads in statement
- Send physical updates and only apply changes
  - ☆ Opposite characteristics than sending statements

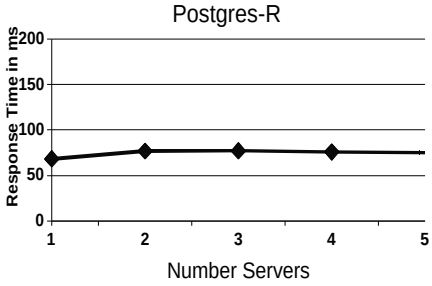
Gain of sending and applying physical changes



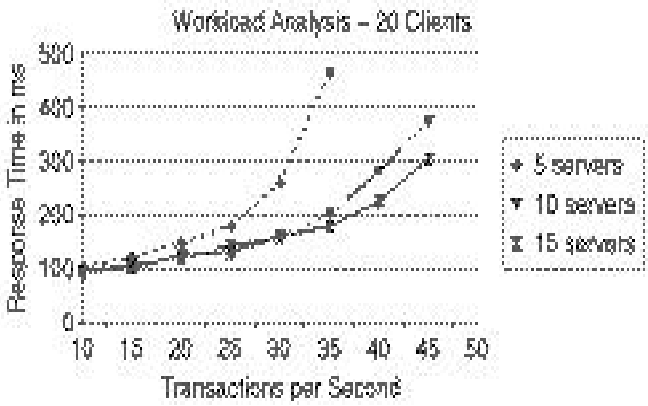
$$Scaleup = \frac{numberOfNodes}{1 + updateRate \; \frac{remoteUpdateCost}{localUpdateCost} \; (numberOfNodes - 1)}$$

Comparison with standard distr. locking

- For all experiments:
  - ☆ Database: 10 relations a 1000 tuples
  - ☆ Transactions: 10 updates per transaction
- Workload: 10 transactions per second
- 5 concurrent clients (each submitting a transaction each 500 milliseconds)

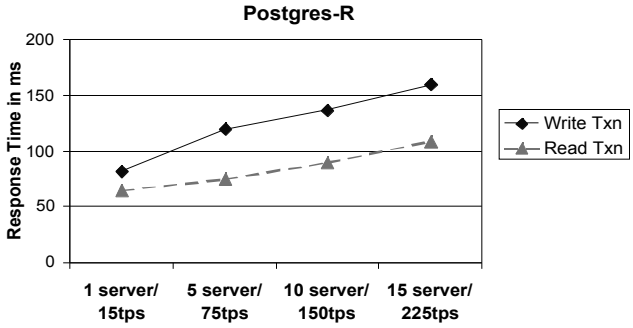


Response Time vs. Throughput



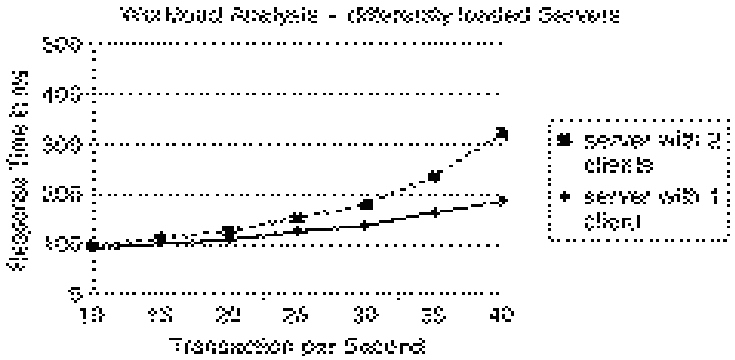
Scalability with fixed workload

- Workload:
  - ☆ 1 update transaction per second per server
  - ☆ 14 queries per second per server
- 3 clients per server



Differently loaded Nodes

- Nodes: 10 nodes
- 15 clients in total



Conclusions

- Eager, update everywhere replication is feasible (at least in clusters) by using adequate techniques
  - ☆ As few messages as possible within transaction boundaries
  - ☆ As few synchronization points as possible
  - ☆ Complete transaction execution only at one site
  - ☆ Simple to adjust to existing concurrency control mechanisms

Current Work

- Recovery under various failure models
- Development of a middleware replication tool
- Development of group communication protocols that better support the needs of the database system
  - ☆ Ordering semantics
  - ☆ Failure models
- Building a complete system
  - ☆ Adding other replica control protocols
  - ☆ System administration
  - ☆ Partial replication functionality