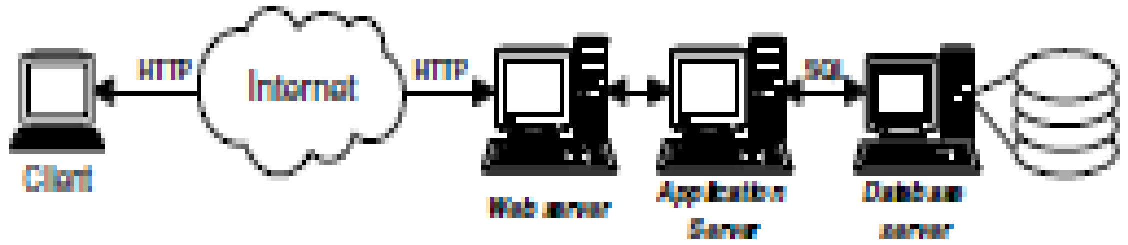
“Distributed Versioning: Consistent Replication for Scaling Back—end Databases of Dynamic Content Web Sites”

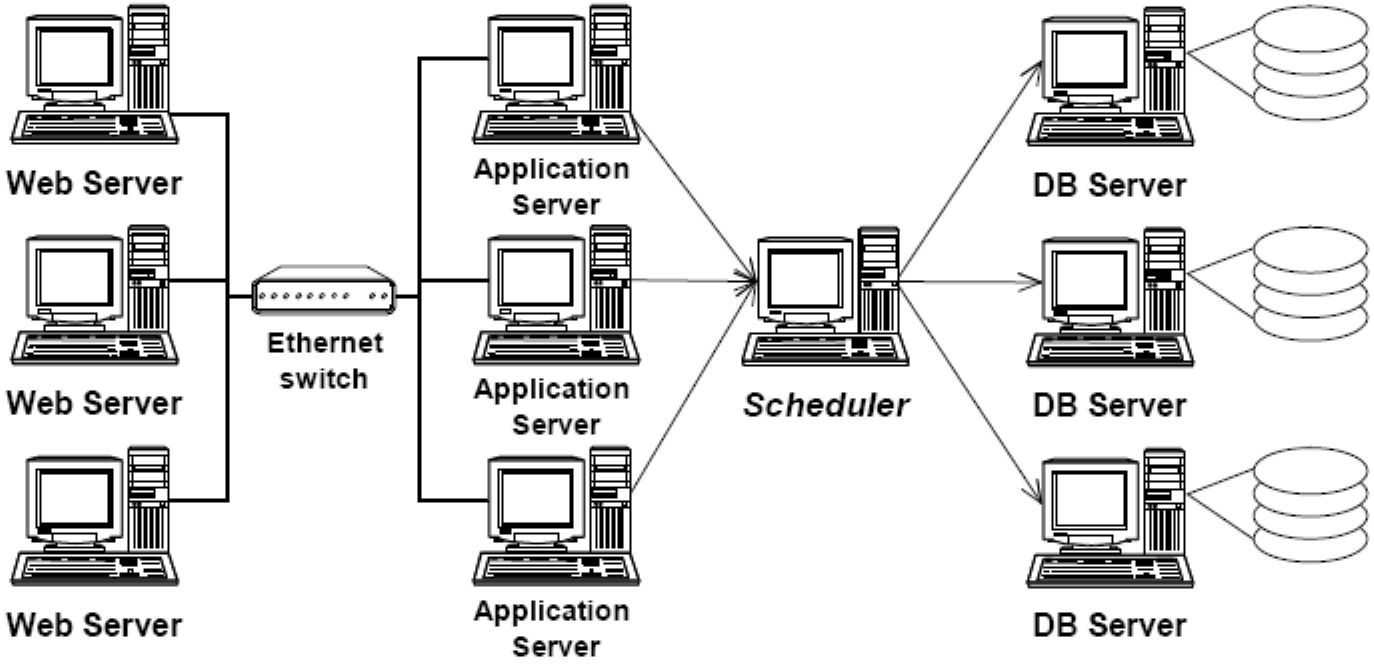
Cristiana Amza, Alan L. Cox and Willy Zwaenepoel.

Proceedings of the ACM/IFIP/Usenix Middleware Conference, June 2003

Distributed Versioning



Distributed Versioning



Distributed versioning

- Each transaction declares the accessed tables and the kind of operation (read/write) before execution.
- Each table has a version number.
- The scheduler assings table versions atomically (one transaction at a time).
  - If two transactions conflict, one will have larger version numbers.
  - Versions are created when a transaction completes its last access to that table.

Distributed versioning

- No version number is assigned to single operation queries (read only). They are forwarded to one replica. It executes after all conflicting transactions complete.
- Other transactions: operations at each replica are executed in version number order. This guarantees that all replicas execute conflicting transactions in the same total order. 1-copy-serializability.

Distributed versioning

- The scheduler sends writes to all replicas. It waits for the first response to reply the client.
- Reads are sent to one replica.
- It maintains for each replica the status of each write operation and the current version number. It sends a read operation that follows a write to a replica that has completed the previous write.

Distributed versioning

```
begin
   write a
   write b
   write c
end
```

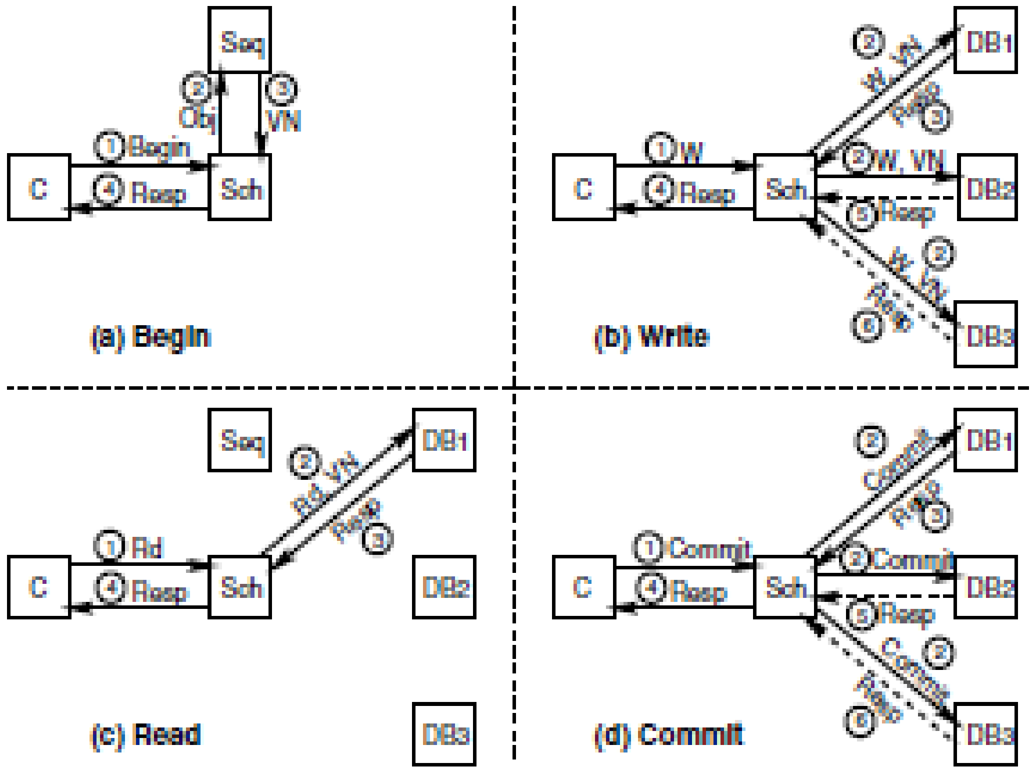
```
         1   2   3   4   5   6
T0:   a0,b0,c0
T1:              a1,b1,c1
```

```
         1   2   3   4
T0:   a0,b0,c0
T1:      a1,b1,c1
```

Distributed versioning. Implementation

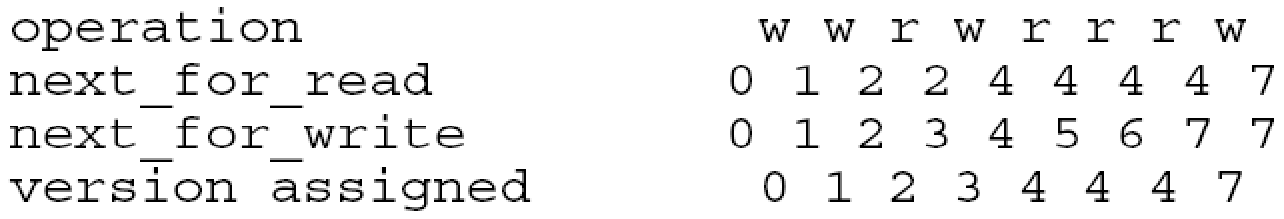
- Three kinds of processes: scheduler (one), sequencer (one) and database proxy (one per replica).
- Transaction start: sequencer assigns version numbers to each accessed table and returns the info to the scheduler.
- The sequencer keeps two values: next-for-read and next-for-write. It returns the corresponding value.

Distributed versioning. Implementation



Distributed versioning. Implementation

- These two counters are incremented when there is a conflicting operation.
- Next-for-write is incremented when there is write and next-for-read is set to next-for-write.
- After a sequence number is assigned for a read operation next-for-write is incremented.



```
operation                   w  w  r  w  r  r  r  w
next_for_read              0  1  2  2  4  4  4  4  7
next_for_write             0  1  2  3  4  5  6  7  7
version assigned            0  1  2  3  4  4  4  7
```

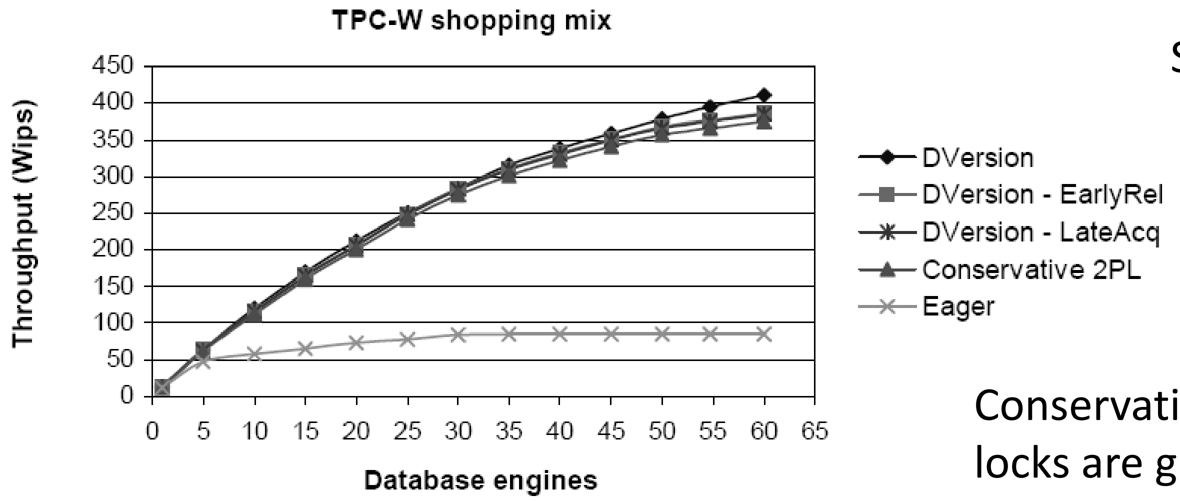
Errors ?

Distributed versioning. Implementation

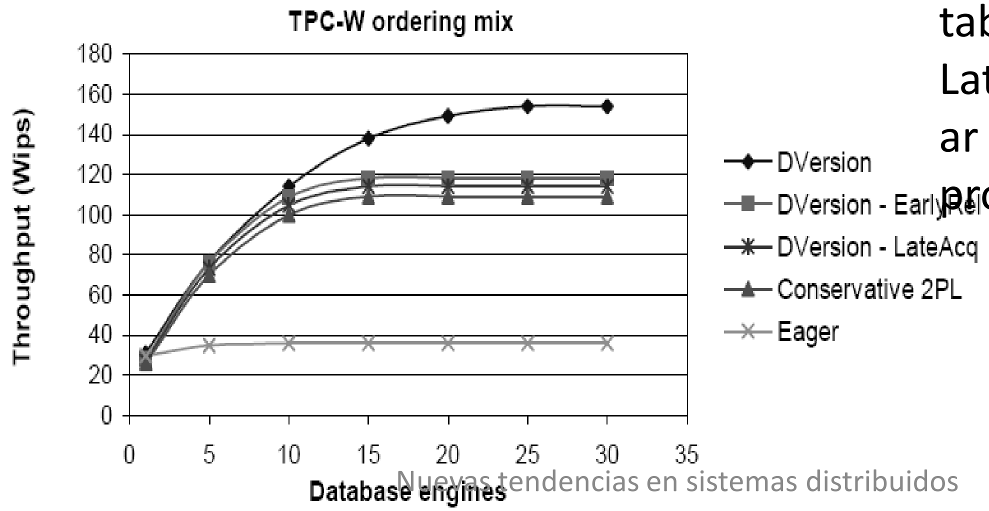
- The DB proxy keeps version numbers.
- A write query is executed at a replica only when the version numbers of each table at the DB match the version numbers of the query.
- A read query is executed when the version numbers are greater than or equal to the version numbers of the query.
- Writes are blocked at the replica and reads by the scheduler.
- Commit/aborts are tagged with version number. It is sent to all replicas. When the tx completes at the DB, the proxy increments the version number of the tables.
- Early version releases: Last-use notation to increment the table version.

operation	w	w	r	w	r	r	r	w
version assigned	0	1	2	3	4	4	4	7
version produced	1	2	3	4	5	6	7	8

Distributed versioning.Performance

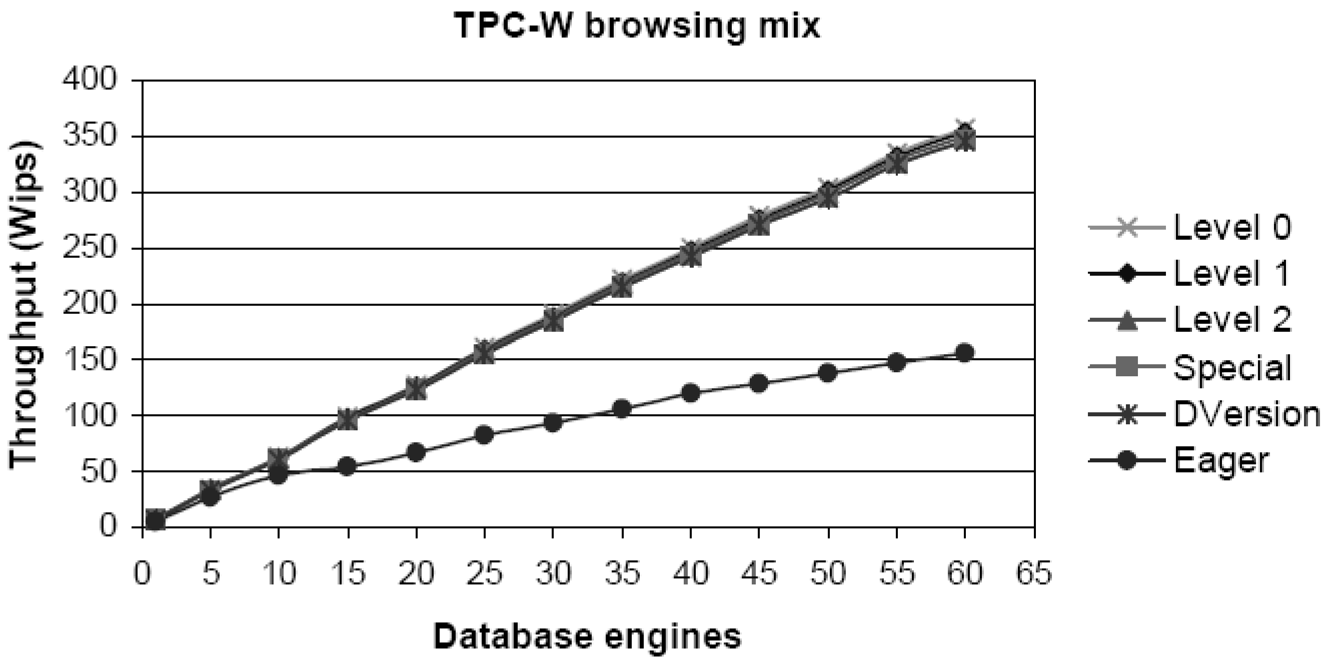


Simulated DB!!



Conservative 2PL: wait until all locks are granted at the begining.
EarlyRel: new versions are produced at commit. Waits for the table version.
LateAcq: Waits for all table versions ar the beginning. Wew versions are produced after last use of a table.

Distributed versioning.Performance



Level 0: lazy update anywhere
Level 1: Writes are totally ordered. Reads maybe inconsistent.
Level 2: Writes are totally ordered. Reads up to x seconds stale, a client reads his/her writes.