

# Database System Internals Replication

Paul G. Allen School of Computer Science and Engineering University of Washington, Seattle

March 10, 2021

CSE 444 - Winter 2020

Ullman Book Chapter 20.6

# Database management systems. Ramakrishnan and Gehrke. Third Ed. Chapter 22.11

- Goals of replication
- Three types of replication
  - Synchronous (aka eager) replication
  - Asynchronous (aka lazy) replication
  - Two-tier replication

# Goals of Replication

# Goal 1: availability

#### Goal 2: performance



• But, it's easy to build a replicated system that reduces performance and availability

# Types of Replication



March 10, 2021

## Synchronous Replication

- Also called eager replication
- All updates are applied to all replicas (or to a majority) as part of a single transaction (need two phase commit)
- Main goal: as if there was only one copy
  - Maintain consistency
  - Maintain one-copy serializability
  - I.e., execution of transactions has same effect as an execution on a non-replicated db
- Transactions must acquire global locks

# Synchronous Replication



## Synchronous Master Replication

#### One master for each object holds primary copy

- The "Master" is also called "Primary"
- To update object, transaction must acquire a lock at the master
- Lock at the master is global lock
- Master propagates updates to replicas synchronously
  - Updates propagate as part of the same distributed transaction
    - Need to run 2PC at the end
  - For example, using triggers



# **Crash Failures**

#### What happens when a secondary crashes?

- Nothing happens
- When secondary recovers, it catches up

#### What happens when the master/primary fails?

- Blocking would hurt availability
- Must chose a new primary: run election

## **Network Failures**

#### Network failures can cause trouble...

- Secondaries think that primary failed
- Secondaries elect a new primary
- But primary can still be running
- Now have two primaries!

To avoid problem, only majority partition can continue processing at any time

#### In general,

- Whenever a replica fails or recovers...
- a set of communicating replicas must determine...
- whether they have a majority before they can continue

# Types of Replication



# Synchronous Group Replication

#### Master-less

- Any node can initiate a transaction!
- Need to gather a number of nodes that agree on a particular transaction



## Synchronous Group Replication

#### With n copies

- Exclusive lock on x copies is global exclusive lock
- Shared lock on s copies is global shared lock
- Must have: 2x > n and s + x > n
- Version numbers serve to identify current copy



# Synchronous Group Replication

#### Majority locking

- $s = x = \lceil (n+1)/2 \rceil$  eg: 11 nodes: need 6 locked
- No need to run any reconfiguration algorithms

#### Read-locks-one, write-locks-all

- s=1 and x = n, high read performance
- Need to make sure algo runs on quorum of computers

## Synchronous Replication Properties

#### Favours consistency over availability

- Only majority partition can process requests
- There appears to be a single copy of the db

#### High runtime overhead

- Must lock and update at least majority of replicas
- Two-phase commit
- Runs at pace of slowest replica in quorum
- So overall system is now slower
- Higher deadlock rate (transactions take longer)

# Types of Replication



#### Asynchronous Replication

- Also called lazy replication
- Also called optimistic replication
- Main goals: availability and performance
- Approach
  - One replica updated by original transaction
  - Updates propagate asynchronously to other replicas

## Asynchronous Replication



#### Asynchronous Master Replication

#### One master holds primary copy

- Transactions update primary copy
- Master asynchronously propagates updates to replicas, which process them in same order (e.g. through log shipping)
- Ensures single-copy serializability
- What happens when master/primary fails?
  - Can lose most recent transactions when primary fails!
  - After electing a new primary, secondaries must agree who is most up-to-date

# Types of Replication





## Asynchronous Group Replication

- Also called multi-master
- Best scheme for availability
- Cannot guarantee one-copy serializability!



#### Asynchronous Group Replication

- Cannot guarantee one-copy serializability!
- Instead guarantee convergence
  - Db state does not reflect any serial execution
  - But all replicas have the same state
- Detect conflicts and reconcile replica states
- Different reconciliation techniques are possible
  - Manual
  - Most recent timestamp wins
  - Site A wins over site B
  - User-defined rules, etc.

## **Detecting Conflicts Using Timestamps**



## **Detecting Conflicts Using Timestamps**



- An extension of Multiversion Concurrency Control (MVCC) to multiple servers
- Standard MVCC: each data item X has a timestamp t: X<sub>4</sub>, X<sub>9</sub>, X<sub>10</sub>, X<sub>14</sub>, ..., X<sub>t</sub>
- Vector Clocks: X has set of [server, timestamp] pairs X([s1,t1], [s2,t2],...)

## Asynchronous Group Replication Properties

- Favours availability over consistency
  - Can read and update any replica
  - High runtime performance
- Weak consistency
  - Conflicts and reconciliation

- Goals of replication
- Three types of replication
  - Synchronous (aka eager) replication
  - Asynchronous (aka lazy) replication
  - Two-tier replication

#### **Two-Tier Replication**

- Benefits of lazy master and lazy group
- Each object has a master with primary copy
- When disconnected from master
  - Secondary can only run tentative transactions
- When reconnects to master
  - Master reprocesses all tentative transactions
  - Checks an acceptance criterion
  - If passes, we now have final commit order
  - Secondary undoes tentative and redoes committed

## Conclusion

- Replication is a very important problem
  - Fault-tolerance (various forms of replication)
  - Caching (lazy master)
  - Warehousing (lazy master)
  - Mobility (two-tier techniques)
- Replication is complex, but basic techniques and trade-offs are very well known
  - Synchronous or asynchronous replication
  - Master or quorum



## Some Popular NewSQL Systems

#### H-Store

- Research system from Brown U., MIT, CMU, and Yale
- Commercialized as VoltDB

#### Hekaton

- Microsoft
- Fully integrated into SQL Server
- Hyper
  - Hybrid OLTP/OLAP
  - Research system from TU Munich. Bought by Tableau
- Spanner
  - Google

### **H-Store Insight**

#### TRADITIONAL DBMS:



#### H-Store Key Ideas

#### Main-memory storage

- Avoids disk IO costs / buffer pool costs
- Durability through snapshots + cmd log
- Replication
- Serial execution
  - One database partition per thread on one core
  - Avoid overheads related to locking
- All transactions are stored procedures
  - Command logging avoids heavy recovery overheads
- Avoid distributed transactions
  - But when needed, run 2PC



#### Voter Benchmark

Japanese "American Idol"



- Focus: DBMS with large main memories and many core CPUs
- Integrated with SQL Server
- Key user-visible features
  - Simply declare a table "memory resident"
  - Hekaton tables are fully durable and transactional, though non-durable tables are also supported
  - Query can touch both Hekaton and regular tables

#### Hekaton Key Details

- Idea: To increase transaction throughput must decrease number of instructions / transaction
- Main-memory DBMS
  - Optimize indexes for memory-resident data
  - Durability by logging and checkpointing records to external storage
- No partitioning
  - Any thread can touch any row of any table
- No locking
  - Uses a new MVCC method for isolation

#### Hekaton More Details

#### Optimized stored procedures

 Compile statements and stored procedures into customized, highly efficient machine code



#### Hybrid OLTP and OLAP

- In-memory data management
  - Including optimized indexes for memory-resident data
  - Data compression for cold data
- Data-centric code generation
  - SQL translated to LLVM
- OLAP separated from OLTP using MVCC
- Exploits hardware transactional memory
- Data shuffling and distribution optimizations

# Conclusion

- Many innovations recently in
  - Big data analytics
  - Transaction processing at very large scale
- Many more problems remain open
- This course teaches foundations
- Innovate with an open mind!