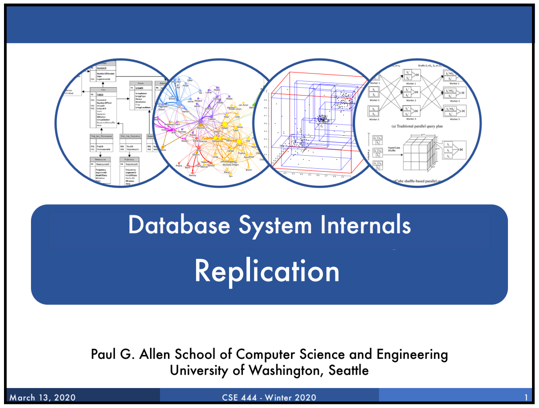
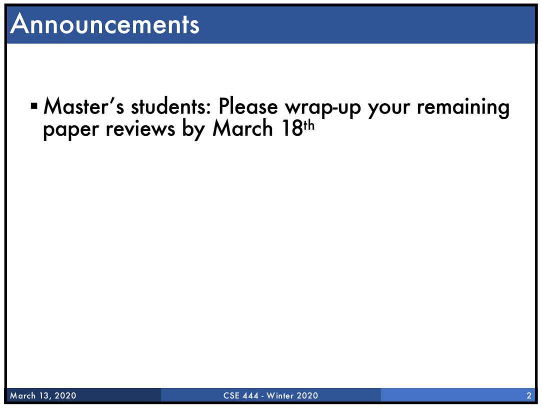


Database System Internals

Replication

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

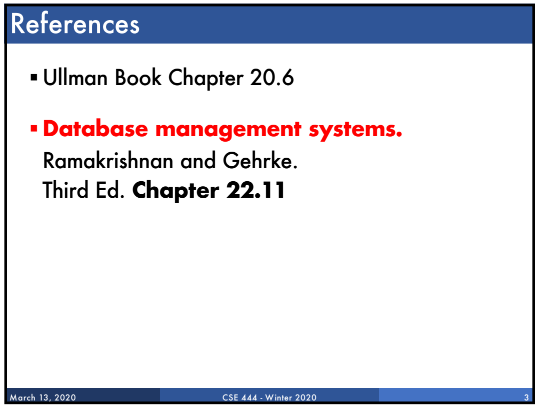
1



Announcements

- Master's students: Please wrap-up your remaining paper reviews by March 18th

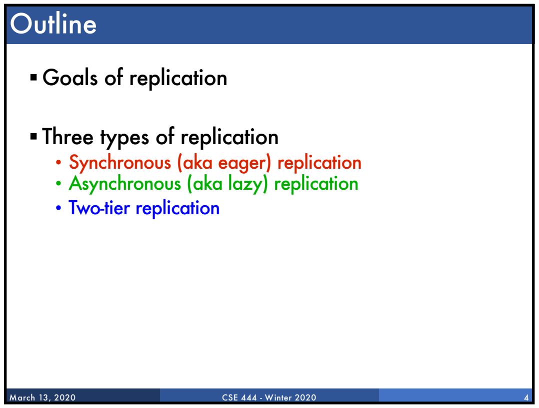
2



References

- Ullman Book Chapter 20.6
- **Database management systems.** Ramakrishnan and Gehrke.
  Third Ed. **Chapter 22.11**

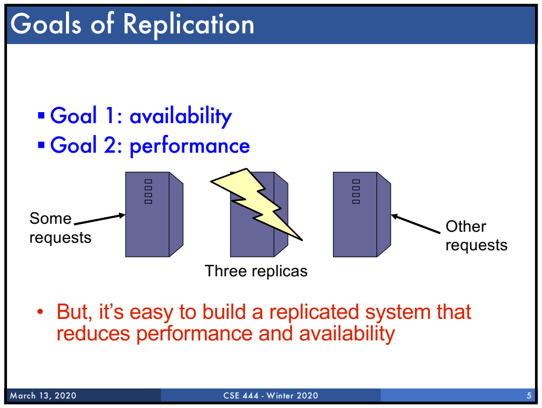
3



Outline

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

4

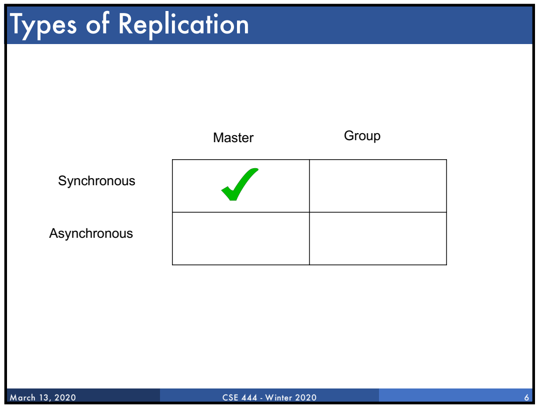


Goals of Replication

- Goal 1: availability
- Goal 2: performance

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

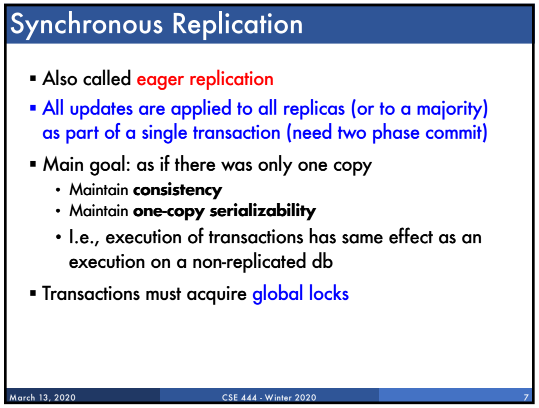
5



Types of Replication

	Master	Group
Synchronous	✓	
Asynchronous		

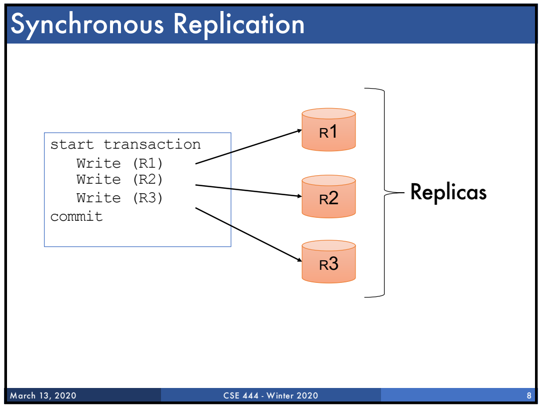
6



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

7



Synchronous Replication

```
start transaction
   Write (R1)
   Write (R2)
   Write (R3)
commit
```

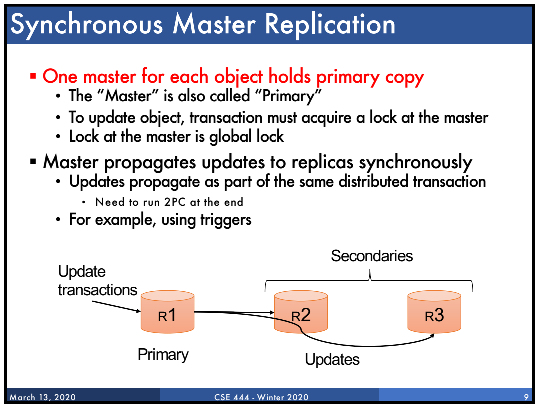
R1

R2

R3

Replicas

8



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

Update transactions

R1

Primary

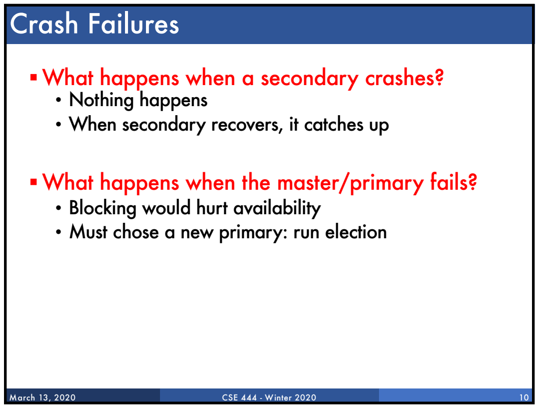
Secondaries

R2

R3

Updates

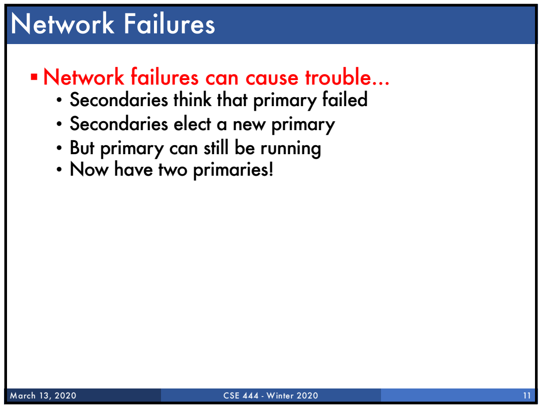
9



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

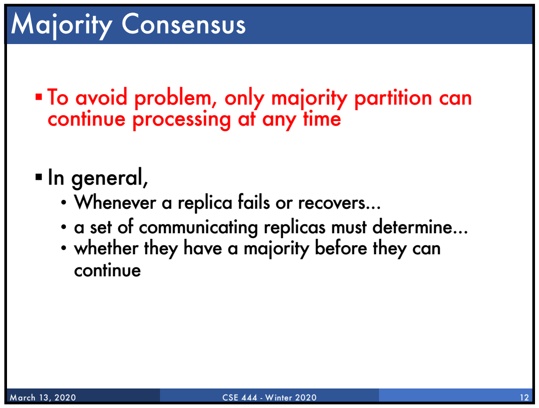
10



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!

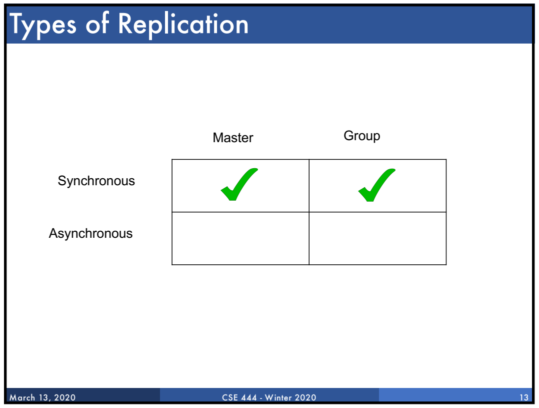
11



Majority Consensus

- 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

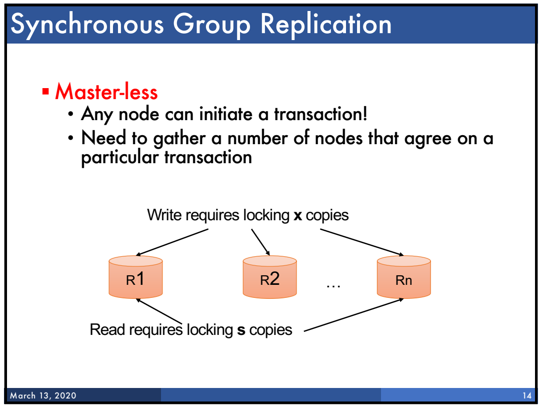
12



Types of Replication

	Master	Group
Synchronous	✓	✓
Asynchronous		

13



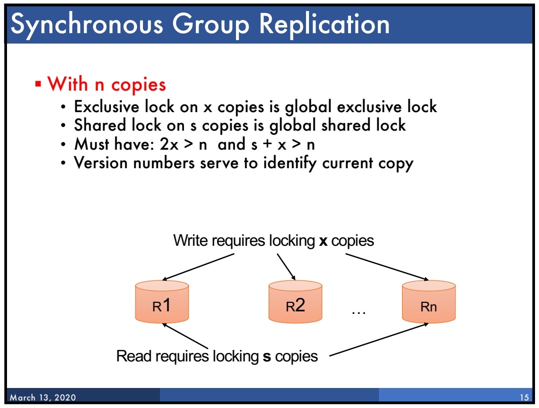
Synchronous Group Replication

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

Write requires locking **x** copies

Read requires locking **s** copies

14



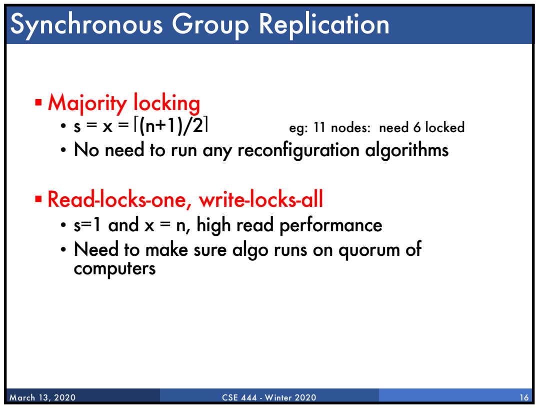
Synchronous Group Replication

- 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

Write requires locking **x** copies

Read requires locking **s** copies

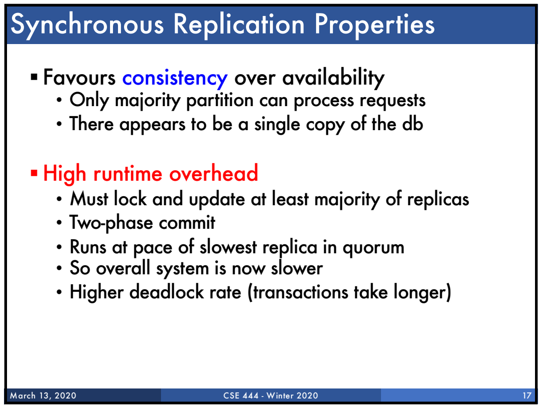
15



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

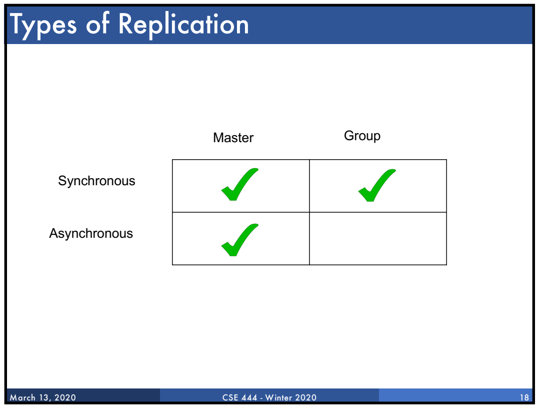
16



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)

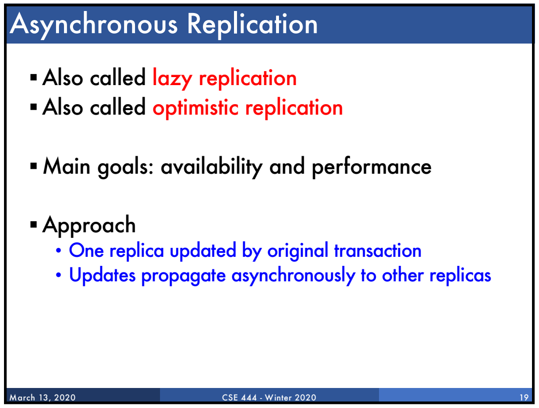
17



Types of Replication

	Master	Group
Synchronous	✓	✓
Asynchronous	✓	

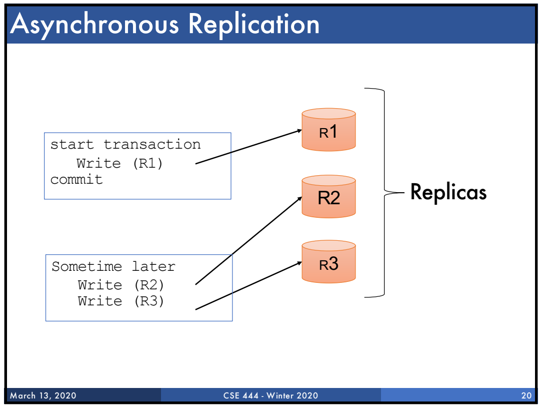
18



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

19

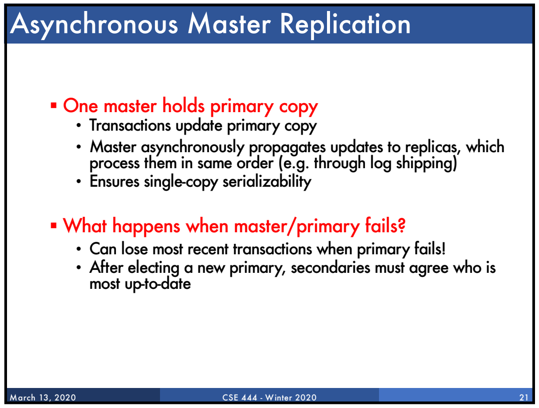


Asynchronous Replication

```
start transaction
   Write (R1)
commit
```

```
Sometime later
   Write (R2)
   Write (R3)
```

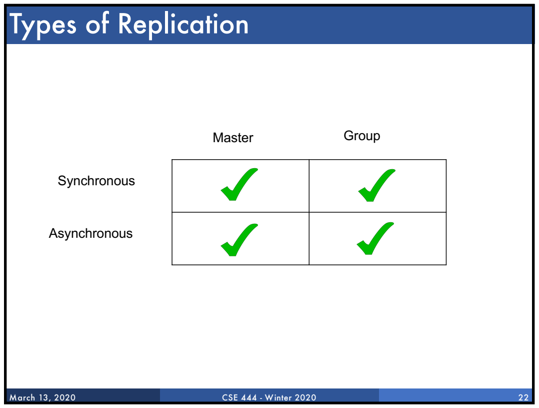
20



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

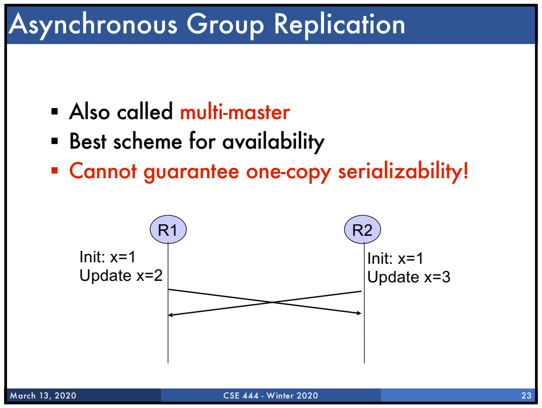
21



Types of Replication

	Master	Group
Synchronous	✓	✓
Asynchronous	✓	✓

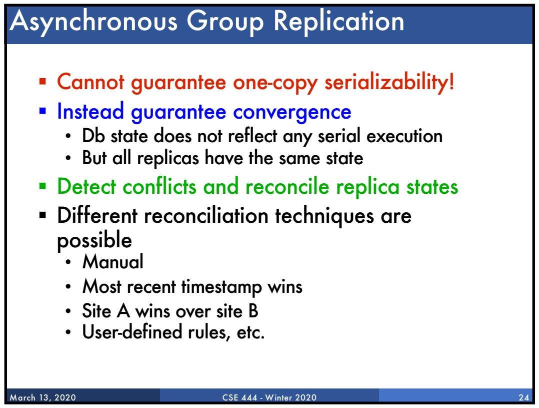
22



Asynchronous Group Replication

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

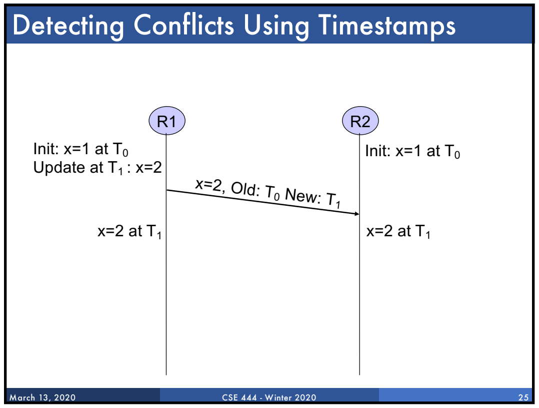
23



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.

24



Detecting Conflicts Using Timestamps

R1

Init: x=1 at T_0
Update at T_1 : x=2

x=2, Old: T_0 New: T_1

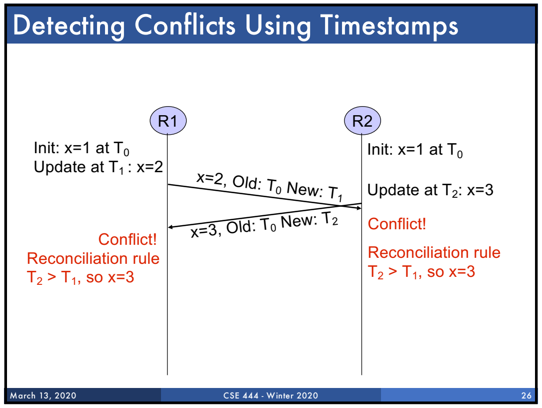
x=2 at T_1

R2

Init: x=1 at T_0

x=2 at T_1

25



Detecting Conflicts Using Timestamps

R1

Init: x=1 at T_0
Update at T_1 : x=2

x=2, Old: T_0 New: T_1

x=3, Old: T_0 New: T_2

Conflict!
Reconciliation rule
$T_2 > T_1$, so x=3

R2

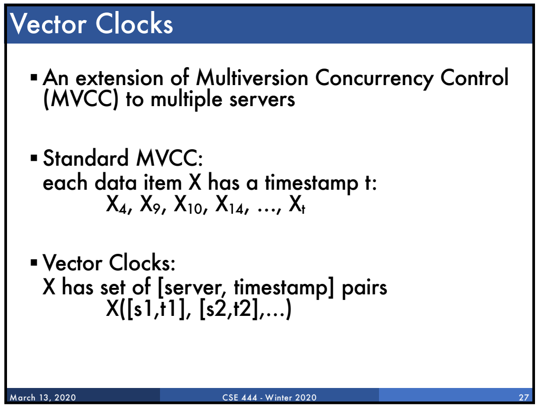
Init: x=1 at T_0

Update at T_2: x=3

Conflict!

Reconciliation rule
$T_2 > T_1$, so x=3

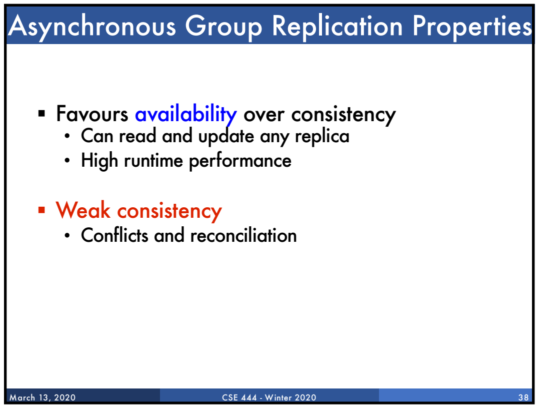
26



Vector Clocks

- An extension of Multiversion Concurrency Control (MVCC) to multiple servers
- Standard MVCC:
  each data item X has a timestamp t:
  $X_4, X_9, X_{10}, X_{14}, \ldots, X_t$
- Vector Clocks:
  X has set of [server, timestamp] pairs
  X([s1,t1], [s2,t2],...)

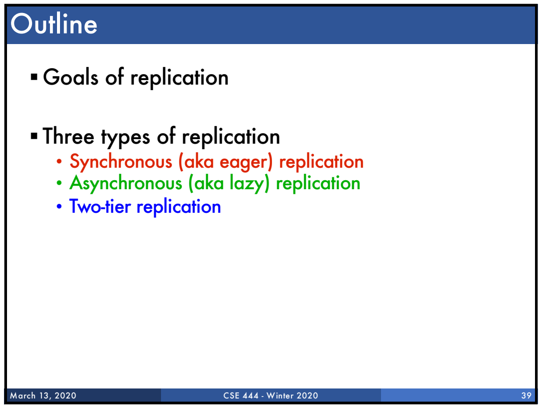
27



Asynchronous Group Replication Properties

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

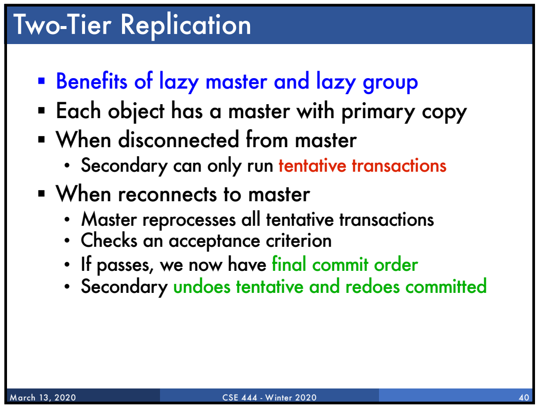
38



Outline

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

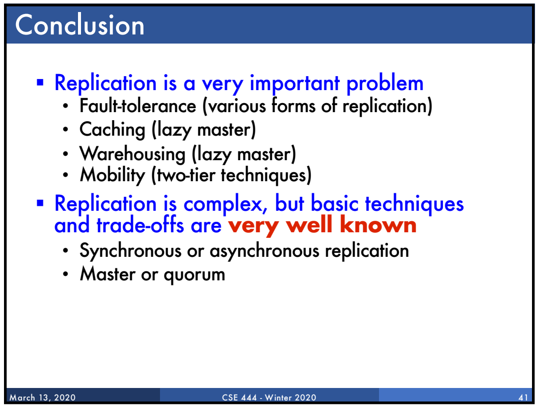
39



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

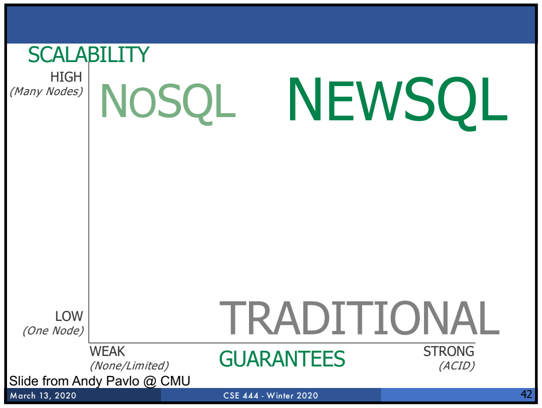
40



Conclusion

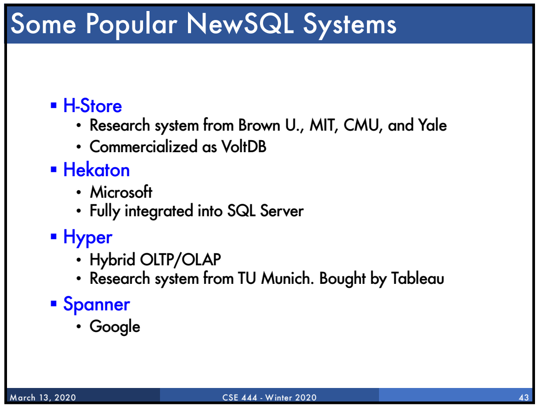
- Replication is a very important problem
  - Fault-tolerance (various forms of replication)
  - Caching (lazy master)
  - Warehousing (lazy master)
  - Mobility (two-tier techniques)
- Replication is complex, but basic techniques and trade-offs are **very well known**
  - Synchronous or asynchronous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

41



Slide from Andy Pavlo @ CMU

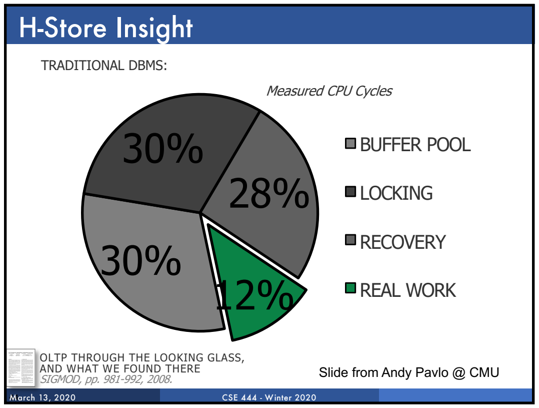
42



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

43

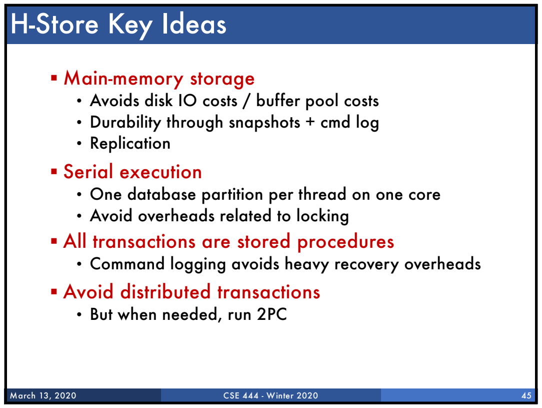


H-Store Insight

OLTP THROUGH THE LOOKING GLASS, AND WHAT WE FOUND THERE
SIGMOD, pp. 981-992, 2008.

Slide from Andy Pavlo @ CMU

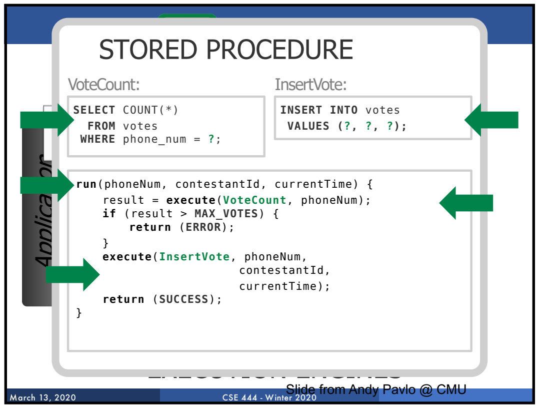
44



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

45



STORED PROCEDURE

VoteCount:

```
SELECT COUNT(*)
  FROM votes
 WHERE phone_num = ?;
```

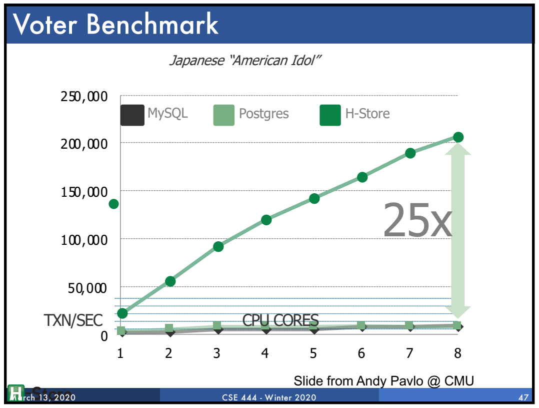
InsertVote:

```
INSERT INTO votes
 VALUES (?, ?, ?);
```

```
run(phoneNum, contestantId, currentTime) {
    result = execute(VoteCount, phoneNum);
    if (result > MAX_VOTES) {
        return (ERROR);
    }
    execute(InsertVote, phoneNum,
                        contestantId,
                        currentTime);
    return (SUCCESS);
}
```

Slide from Andy Pavlo @ CMU

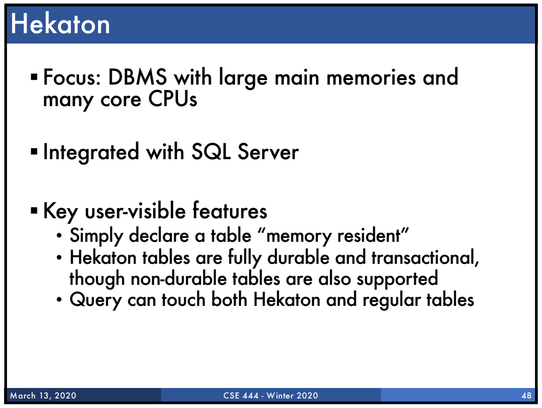
46



Voter Benchmark

Slide from Andy Pavlo @ CMU

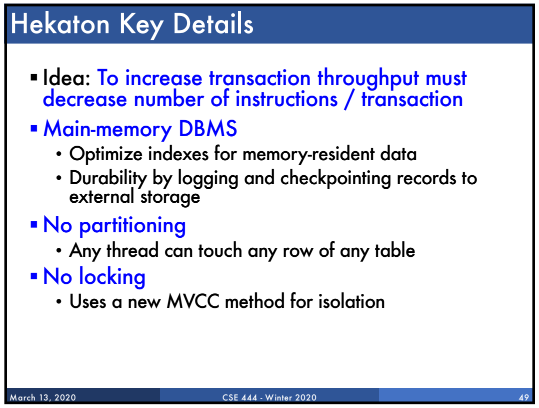
47



Hekaton

- 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

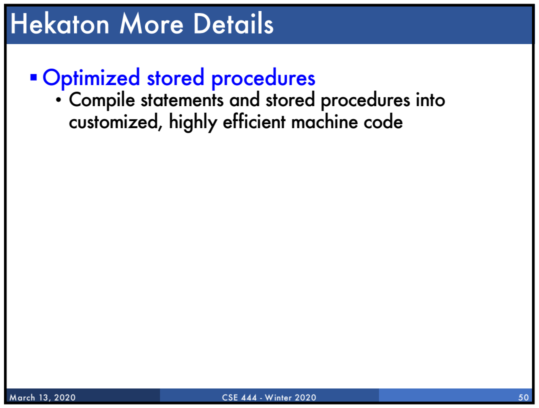
48



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

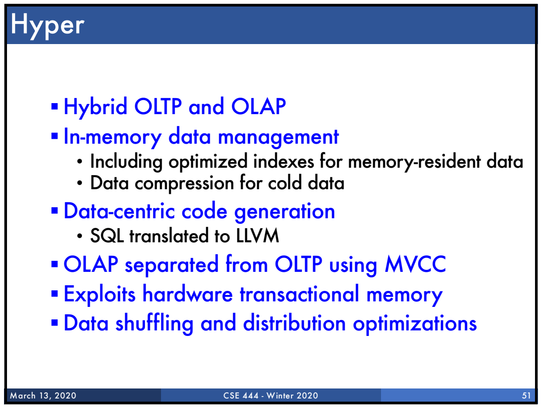
49



Hekaton More Details

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

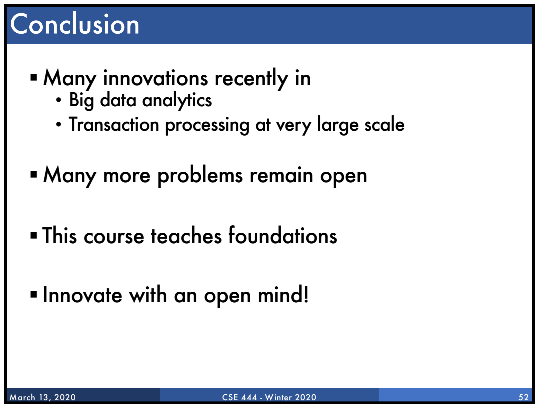
50



Hyper

- 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

51



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!

52