Database Configuration, Tuning, Replication, and Sequence Mining

Dennis Shasha

Courant Institute of Mathematical Sciences

Department of Computer Science

New York University

shasha@cs.nyu.edu

http://cs.nyu.edu/cs/faculty/shasha/index.html

occasional database tuning consultant for Wall Street, Bells, and Pharmas

Overview

- Configuration — disaster-proof systems, interoperability among different languages and different databases.

- Tuning — clustering, concurrency, and hashing; forcing plans.

- Mining time.

Preparing for Disaster

- Database instructors often assume that disks are stable for all time. Not true.
- Memory fails, disks fail (in batches), fires happen (Credit Lyonnais, NY Stock Exchange), and power grids fail. If your system is still alive, you have a big advantage.
- You can even let your competitors use your facilities ... for a price.

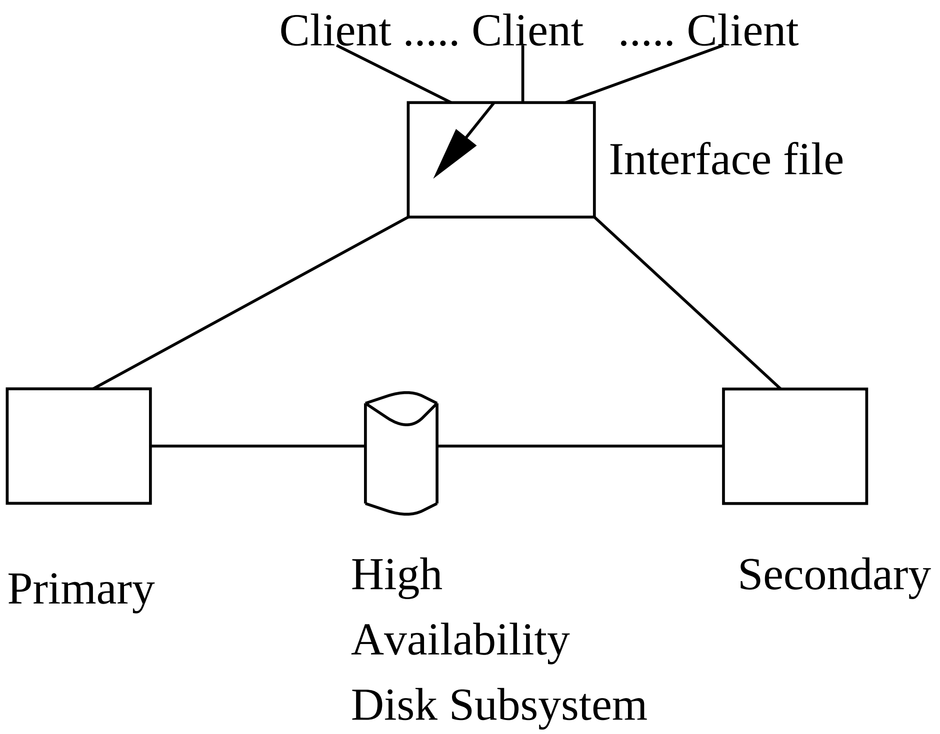
Case: Bond Futures

- Server for trading bond futures having to do with home mortgages.

- Application used only a few days per month, but the load is heavy. During a weekend batch run, 11 out of 12 two-gigabyte disks from a single vendor-batch failed.

High Availability Servers

- A pair of shared memory multiprocessors attached to RAID disks.
- If the primary multiprocessor fails, the backup does a warm start from the disks.
- If a disk fails, RAID masks it.
- Does not survive disasters or correlated failures.



Writes go to the primary and into the high availability disk subsystem. This subsystem is normally a RAID device, so can survive one or more disk failures.
If the primary fails, the secondary works off the same disk image (warm start recovery).

Vulnerability: High availability disk subsystem fails entirely.

Dump and Load

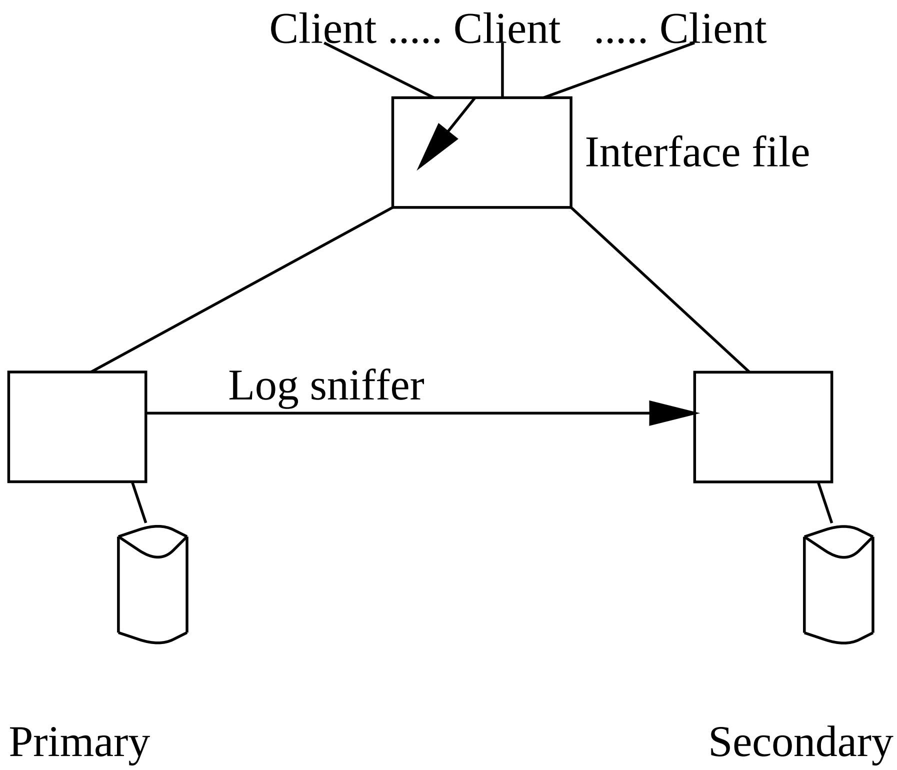
- Full dump at night. Incremental dumps every three minutes.
- Can lose committed transactions, but there is usually a paper trail.
- Backup can be far away: fewer correlated failures.

Replication Server

- Full dump nightly. All operations at the primary are sent to the secondary after commit on the primary.

- May lose a few seconds of committed transactions.

- Major pain to administer, e.g. schemas, triggers, log size.



Basic architecture of a replication server.
The backup reads operations after they are committed on the primary. Upon failure, the secondary becomes the primary by changing the interface file configuration variables.

Vulnerability: if there is a failure of the primary after commit at the primary but before the data reaches the secondary, we have trouble.

Remote Mirroring

- Writes to local disks are mirrored to disks on a remote site. The commit at the local machine is delayed until the remote disks respond.

- Backup problems may cause primary to halt.

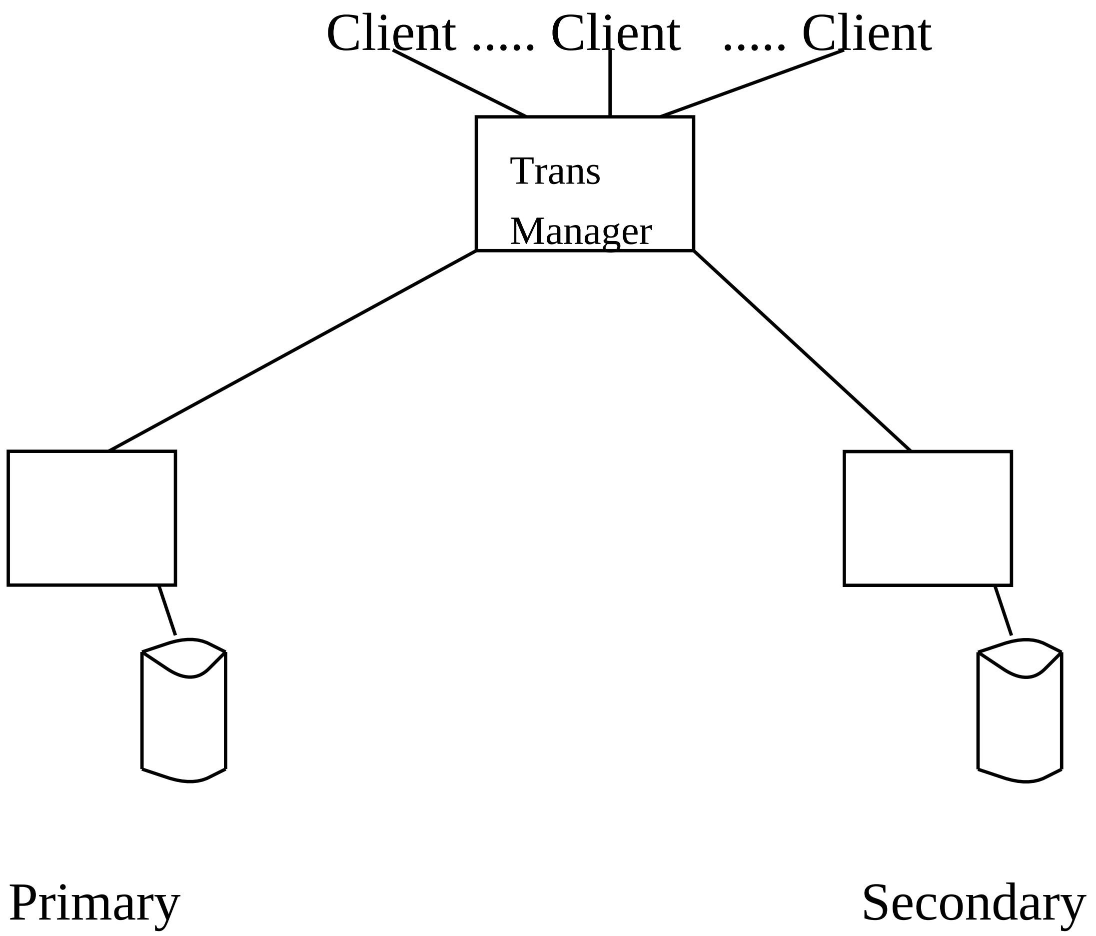
- Reliable buffering can be used (e.g. Qualix), but the net result is rep server without the ability to query the backup.

Two Phase Commit

- Commits are coordinated between the primary and backup.

- Blocking can occur if the transaction monitor fails. Delays occur if backup is slow.

- Wall Street is scared of this.



Two phase commit: transaction manager ensures that updates on the primary and secondary are commit-consistent. This ensures that the two sides are in synchrony.

Vulnerability: blocking or long delays may occur at the primary either due to delays at the secondary (in voting) or failure of the transaction manager.

Quorum Approach (e.g., DEC, HP, IBM, ISIS....)

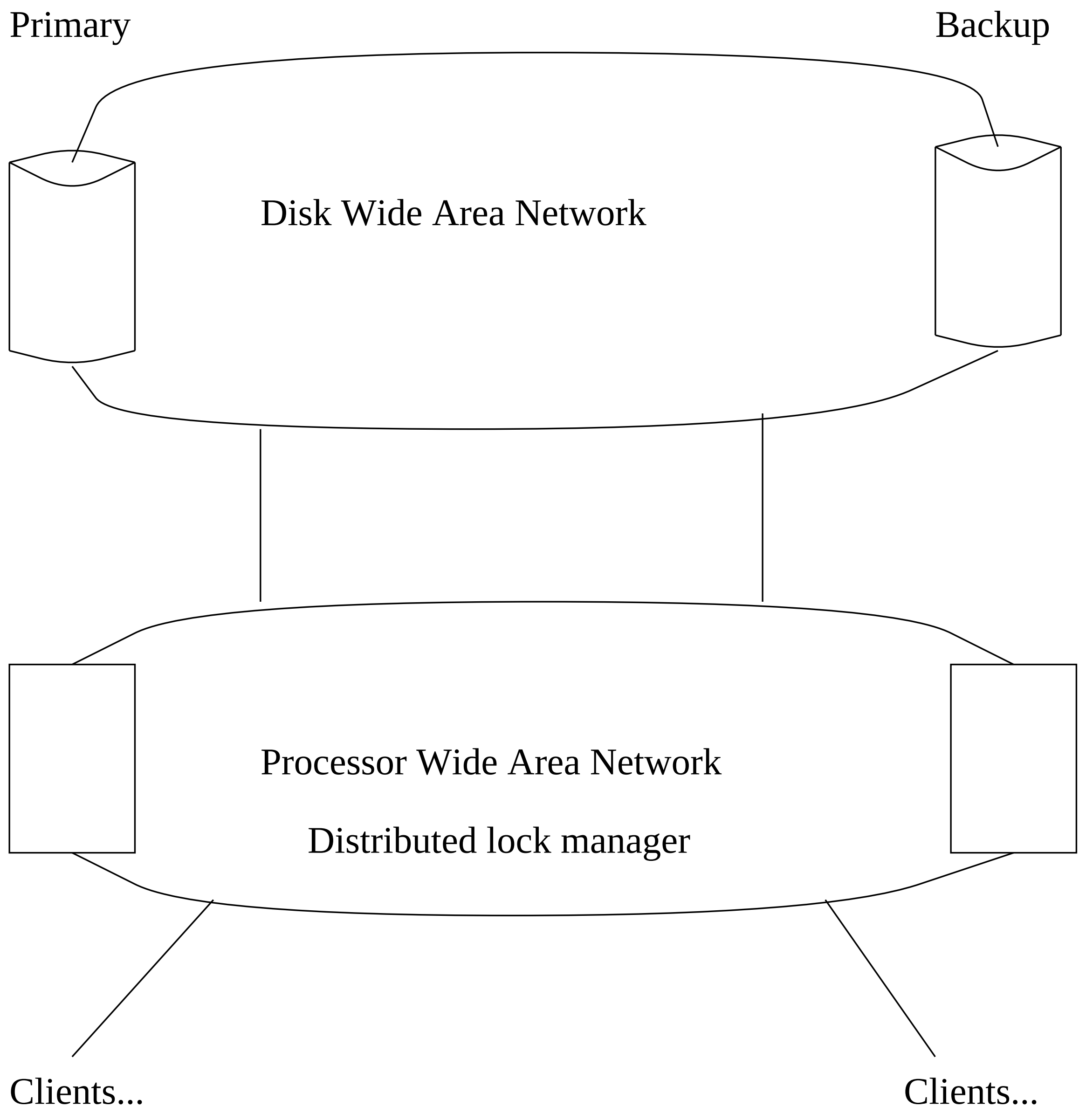
- Servers are co-equal and are interconnected via a highly redundant wide area cable.

- Clients can be connected to any server. Servers coordinate via a distributed lock manager.

- Disks are connected with the servers at several points and to one another by a second wide area link.

Heartbeats

- Heartbeats monitor the connectivity among the various disks and processors.
- If a break is detected, one partition holding a majority of votes continues to execute.
- Any single failure of a processor, disk, site, or network is invisible to the end users (except for a loss in performance).



Quorum Approach as Used in most Stock and Currency Exchanges.
Survives Processor, Disk, and Site failures.

Quorum approach used in most exchanges.

Which to Use

- Stock exchanges use the quorum approach.

- Midoffice database servers often use dump and load or rep server, because they don't mind losing a few trades (traders are supposed to remember).

- Doubleclick: use something simple and that makes you independent of local powergrid failures.

Case: Interdatabase Communication

- As in many industries, financial database systems grow up as islands and then discover — surprise — they must interconnect. Source sends data to destination which then sends some confirmation.

- Replication server is a possible solution to this problem, but
  (i) Commits at the source may not make it.
  (ii) Responses from the destination imply two-way replication. Known to be hazardous.
  (iii) Hard to determine where the data is.

Use Tables as Buffers

- Implement a buffer table on the source system side that holds information in the denormalized form required by the destination side.

- The destination database reads a new tuple t from the buffer table.

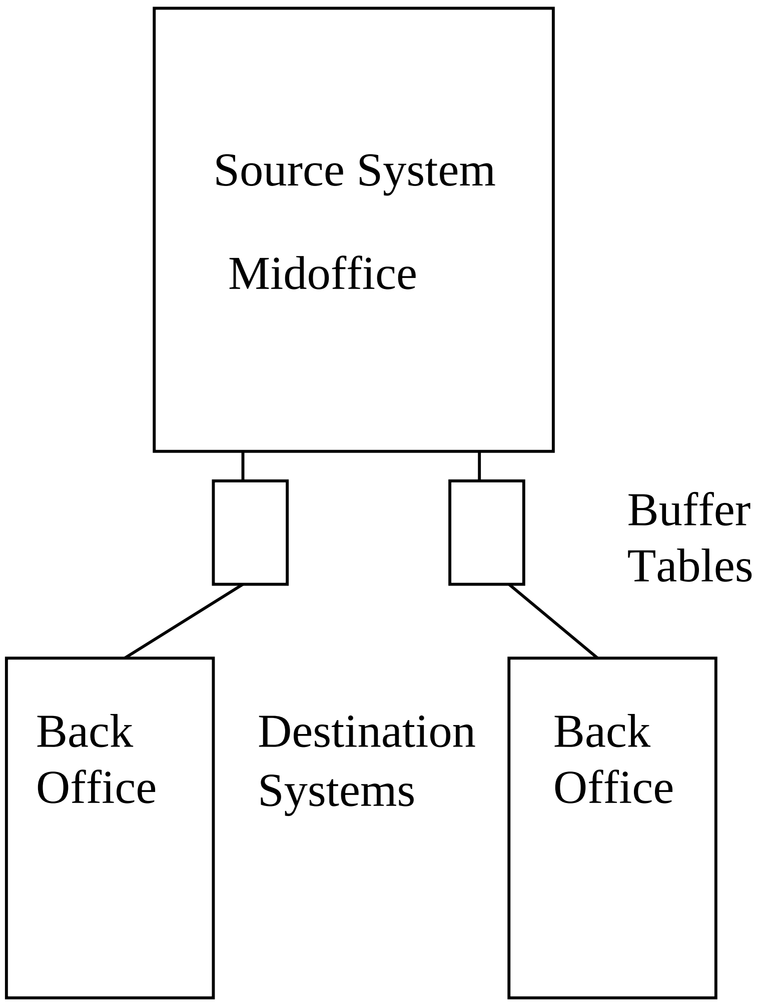
- After processing t, the destination database flags t as deletable or deletes t itself in the buffer table.

Overcoming Blocking

- Get blocking if source and destination scan the buffer table for update purposes.

- Approach 1: Destination database puts responses in a different table to avoid update-update conflicts on the buffer table.

- Approach 2: Use clustering in the buffer to avoid blocking.



Source system transactions write to buffer tables and back office systems read from them.
If back office systems must respond, then either cluster the buffer tables or use a second response table written by the back office system.

Buffer Tables

Case: Batch and Global Trading

- When the trading day is over, there are many operations that must be done to move trades to the backoffice, to clear out positions that have fallen to zero and so on. Call it "rollover."

- Straightforward provided no trades are hitting the database at the same time.

- In a global trading situation, however, rollover in New York may interfere with trading in Tokyo.

Chop the batch

- "Chop" the rollover transaction into smaller ones.
- The conditions for chopping are that the ongoing trades should not create conflict cycles with the rollover pieces.
- New trades don't conflict with rollover. Lock conflicts are due to the fact that rollover uses scans.

Good Candidates for Chopping

- Batch operations that don't logically conflict with ongoing operations. (Index conflicts are not a problem).

- Chopping means take each batch operation and break it into independent pieces, e.g., delete zero-valued positions, update profit and loss.

- If batch operations are not idempotent, it is necessary to use a "breadcrumb" table that keeps track of which batch operations a process has completed.

Tuning Case: Sequential keys, clustering and blocking

- Sequential keys (i.e., keys whose values are monotonic in time) are used to identify rows in trade and position tables uniquely. Suppose the table is clustered on a sequential key.

- Buffer behavior is good since all inserts hit the same few pages.

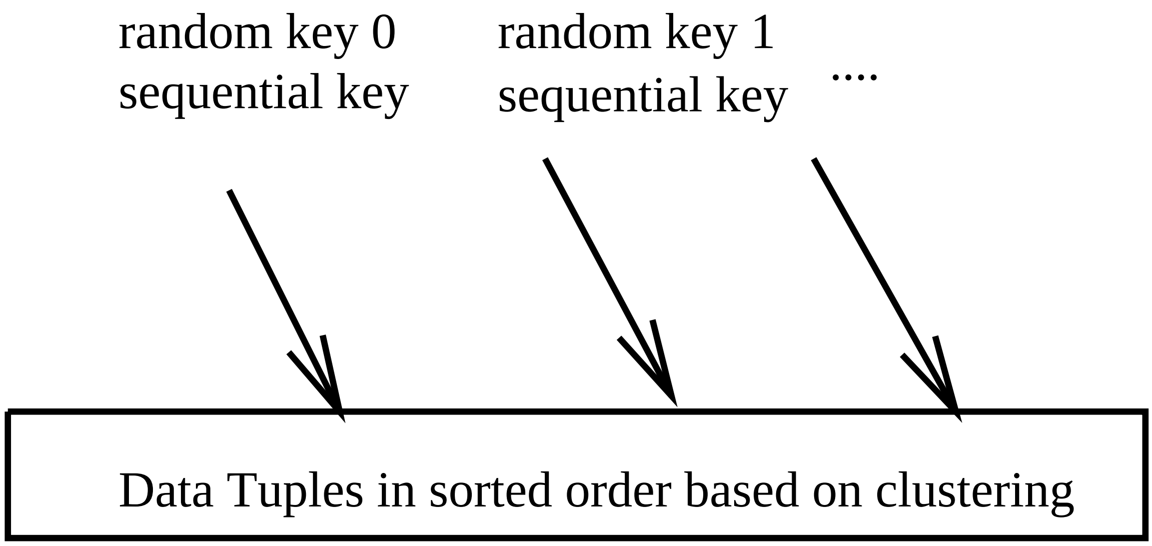
- Multiple concurrent inserts will conflict on the last page of a data structure or of a data page. Especially bad for page-level locking systems.

Hash Clusters

- Create a key:
  concat(hash(process id), sequential key).

- inserts cluster at as many locations as there are possible hash values.

- Good clustering without concurrency loss.



Different random key, sequential key concatenations will not conflict with one another.
They will still however give good buffering behavior since only one page per random key need be in the database cache.

RAID disks

- Raid 5 discs seem to be much faster for read applications than Raid 0, giving approximately a 60% improvement (14 minutes to 5 minutes)

- Raid 5: each disk has a portion of a sector. there are n subsector portion and a parity subsector that make up a sector. A small update will write a single subsector.

RAIDs on online transaction processing

- How Raid 5 works when updating a subsector: it must read the old version of that subsector `Sold`, read the parity subsector `Pold`, `Pnew := (Sold xor Snew) xor Pold`

- This is two reads and two writes for one write. So, Raid 5 is MUCH WORSE for writing applications than Raid 0 or simple mirroring.

- Savage and Wilkes (Usenix ref) have a nice solution to this problem that involves delaying the write to the parity disk.

References for first part

1. The Dangers of Replication and a Solution. Jim Gray, Pat Helland, Pat O'Neil, and Dennis Shasha, Sigmod 1996. pp. 173-182

2. Is GUI Programming a Database Research Problem? Nita Goyal, Charles Hoch, Ravi Krishnamurthy, Brian Meckler, and Michael Suckow. Sigmod 1996, p. 517-528.

3. The COMFORT Automatic Tuning Project Gerhard Weikum, Christof Hasse, Axel Moenkeberg, and Peter Zabback. Information Systems, vol. 19 no. 5, July 1994.

4. Transaction Chopping: Algorithms and Performance Studies. Dennis Shasha, Francois Llirbat, Eric Simon, Patrick Valduriez *ACM Transactions on Database Systems*, October 1995, pp. 325-363.

References for first part – Continued

5. "AFRAID — A frequently redundant array of independent disks" Stefan Savage and John Wilkes 1996 USENIX Technical Conference, January 22-26, 1996

6. *Database Tuning: a principled approach* Prentice-Hall, 1992. (Dennis Shasha) The book that got me all these gigs.