Log Record Buffering

Log records are buffered

Output when a block of log records is full or a **log force** operation

Log force is performed to commit a transaction

A single output operation for several log records

Log Record Buffering

Log records are buffered

Output when a block of log records is full or a **log force** operation

Log force is performed to commit a transaction

A single output operation for several log records

The following rules must be followed

Log records are output to stable storage in the order in which they are created

Transaction T_i enters the commit state only when the log record <T_i **commit**> has been output to stable storage

Write-ahead logging or **WAL** rule to ensure undo information of output records

KNOWLEDGE & SOFTWARE ENGINEERING

Database Buffering

an in-memory buffer of data blocks

the **no-force policy**

the **steal policy**

write ahead logging

no updates while output to disk

Database Buffering

- an in-memory buffer of data blocks
- the **no-force policy**
- the **steal policy**
- write ahead logging
- no updates while output to disk

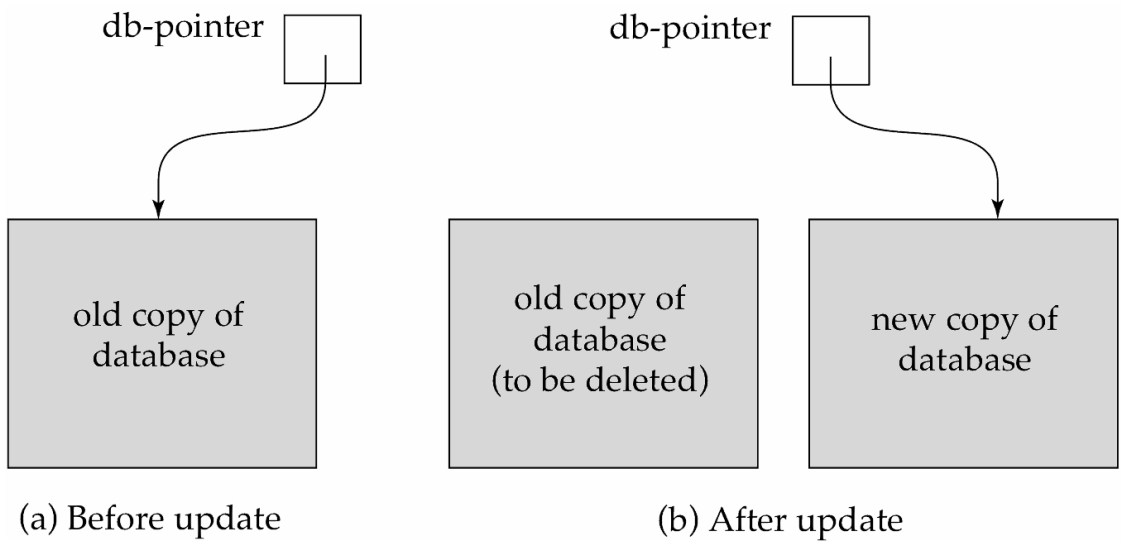
To output a block to disk

1. Acquire an exclusive latch on the block
   - Ensures no update can be in progress on the block
2. Perform a **log flush**
3. Output the block to disk
4. Release the latch on the block

KNOWLEDGE & SOFTWARE ENGINEERING

Shadow paging

- Less used alternative: **shadow-copy** and **shadow-paging**



shadow-copy

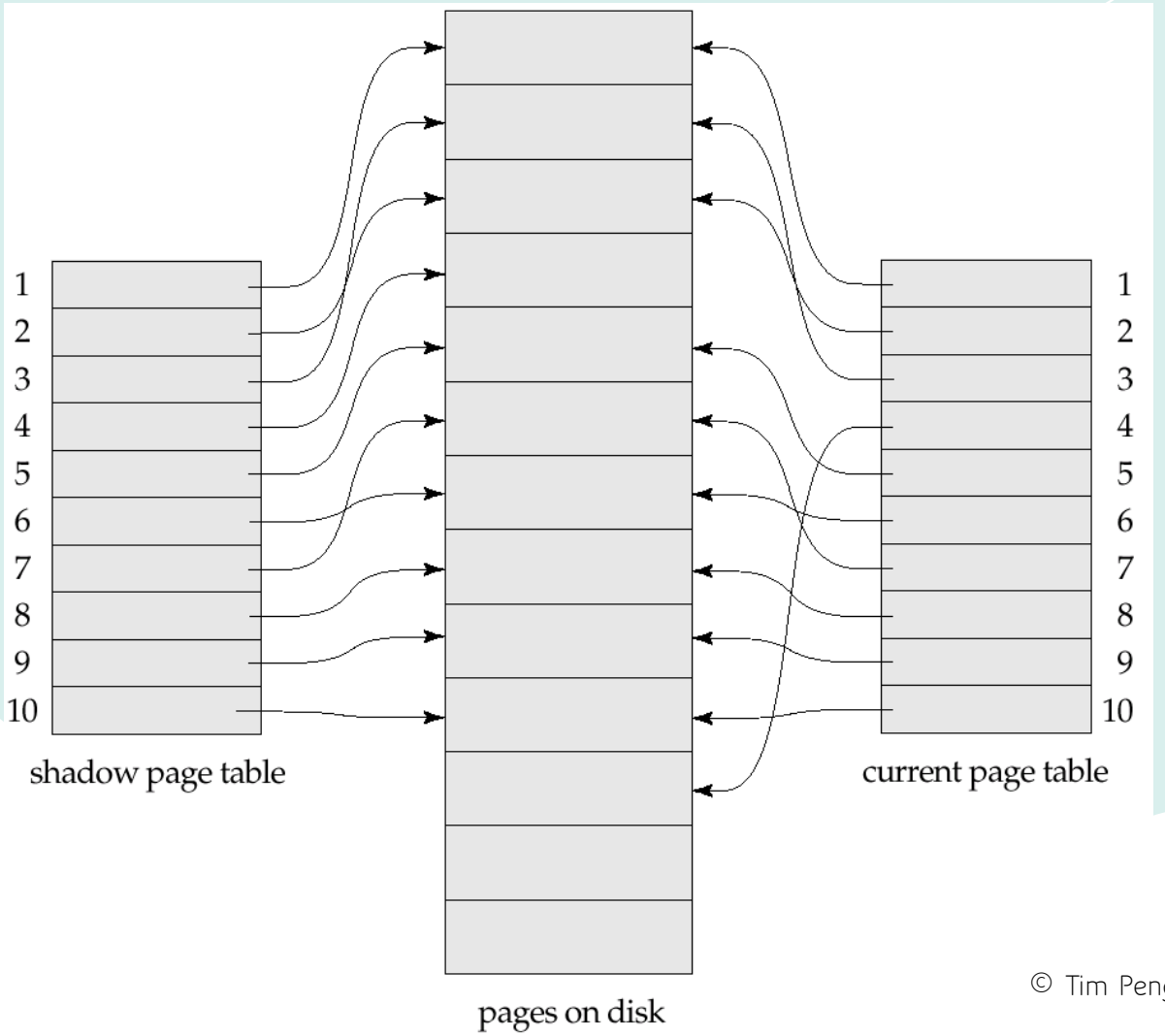
KNOWLEDGE & SOFTWARE ENGINEERING

Shadow Paging

- **Shadow paging** is an alternative to log-based recovery; this scheme is useful if transactions execute serially
- Idea: maintain *two* page tables during the lifetime of a transaction –the **current page table**, and the **shadow page table**
- Store the shadow page table in nonvolatile storage, such that state of the database prior to transaction execution may be recovered.
  - Shadow page table is never modified during execution
- To start with, both the page tables are identical. Only current page table is used for data item accesses during execution of the transaction.
- Whenever any page is about to be written for the first time
  - A copy of this page is made onto an unused page.
  - The current page table is then made to point to the copy
  - The update is performed on the copy

KNOWLEDGE & SOFTWARE ENGINEERING

Shadow and current page tables after write to page 4



KNOWLEDGE & SOFTWARE ENGINEERING

Shadow Paging (Cont.)

- To commit a transaction :
  1. Flush all modified pages in main memory to disk
  2. Output current page table to disk
  3. Make the current page table the new shadow page table, as follows:
     - keep a pointer to the shadow page table at a fixed (known) location on disk.
     - to make the current page table the new shadow page table, simply update the pointer to point to current page table on disk
- Once pointer to shadow page table has been written, transaction is committed.
- No recovery is needed after a crash — new transactions can start right away, using the shadow page table.
- Pages not pointed to from current/shadow page table should be freed (garbage collected).

Shadow Paging (Cont.)

- Advantages of shadow-paging over log-based schemes
  - no overhead of writing log records
  - recovery is trivial
- Disadvantages:
  - Copying the entire page table is very expensive
    - Can be reduced by using a page table structured like a B^+-tree
      - No need to copy entire tree, only need to copy paths in the tree that lead to updated leaf nodes
  - Commit overhead is high even with above extension
    - Need to flush every updated page, and page table
  - Data gets fragmented (related pages get separated on disk)
  - After every transaction completion, the database pages containing old versions of modified data need to be garbage collected
  - Hard to extend algorithm to allow transactions to run concurrently
    - Easier to extend log based schemes

KNOWLEDGE & SOFTWARE ENGINEERING

Failure with Loss of Nonvolatile Storage

So far we assumed no loss of non-volatile storage
Technique similar to checkpointing used to deal with loss of non-volatile storage

Periodically **dump** the entire content of the database to stable storage

No transaction may be active during the dump procedure; a procedure similar to checkpointing must take place

- Output all log records currently residing in main memory onto stable storage.
- Output all buffer blocks onto the disk.
- Copy the contents of the database to stable storage.
- Output a record <**dump**> to log on stable storage.

Failure with Loss of Nonvolatile

Periodically **dump** the entire content of the database to stable storage

No transaction may be active during the dump procedure; a procedure similar to checkpointing must take place

- Output all log records currently residing in main memory onto stable storage.
- Output all buffer blocks onto the disk.
- Copy the contents of the database to stable storage.
- Output a record <**dump**> to log on stable storage.

To recover from disk failure

- restore database from most recent dump.
- Consult the log and redo all transactions that committed after the dump

Can be extended to allow transactions to be active during dump; known as **fuzzy dump** or **online dump**

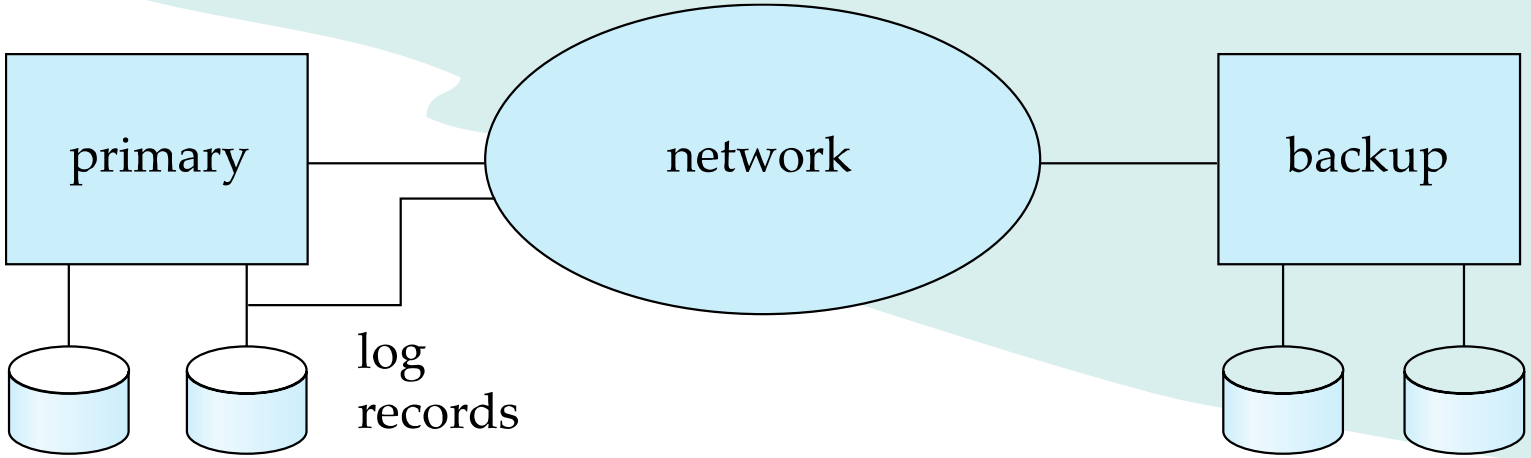
Similar to fuzzy checkpointing

KNOWLEDGE & SOFTWARE ENGINEERING

Remote Backup Systems

Remote Backup Systems

Remote backup systems provide high availability by allowing transaction processing to continue even if the primary site is destroyed.



Remote Backup Systems

Detection of failure: Backup site must detect when primary site has failed

- to distinguish primary site failure from link failure maintain several communication links between the primary and the remote backup.
- Heart-beat messages

Transfer of control:

- To take over control backup site first perform recovery using its copy of the database and all the log records it has received from the primary.
  - Thus, completed transactions are redone and incomplete transactions are rolled back.
- When the backup site takes over processing it becomes the new primary
- To transfer control back to old primary when it recovers, old primary must receive redo logs from the old backup and apply all updates locally.

KNOWLEDGE & SOFTWARE ENGINEERING

Remote Backup Systems (Cont.)

Time to recover: To reduce delay in takeover, backup site periodically process the redo log records (in effect, performing recovery from previous database state), performs a checkpoint, and can then delete earlier parts of the log.

Hot-Spare configuration permits very fast takeover:

- Backup continually processes redo log record as they arrive, applying the updates locally.
- When failure of the primary is detected the backup rolls back incomplete transactions, and is ready to process new transactions.

Alternative to remote backup: distributed database with replicated data
Remote backup is faster and cheaper, but less tolerant to failure

Remote Backup Systems (Cont.)

- Ensure durability of updates by delaying transaction commit until update is logged at backup; avoid this delay by permitting lower degrees of durability.
- **One-safe**: commit as soon as transaction's commit log record is written at primary
  - Problem: updates may not arrive at backup before it takes over.
- **Two-very-safe**: commit when transaction's commit log record is written at primary and backup
  - Reduces availability since transactions cannot commit if either site fails.
- **Two-safe**: proceed as in two-very-safe if both primary and backup are active. If only the primary is active, the transaction commits as soon as is commit log record is written at the primary.
  - Better availability than two-very-safe; avoids problem of lost transactions in one-safe.

KNOWLEDGE & SOFTWARE ENGINEERING

End of Recovery System

KNOWLEDGE & SOFTWARE ENGINEERING