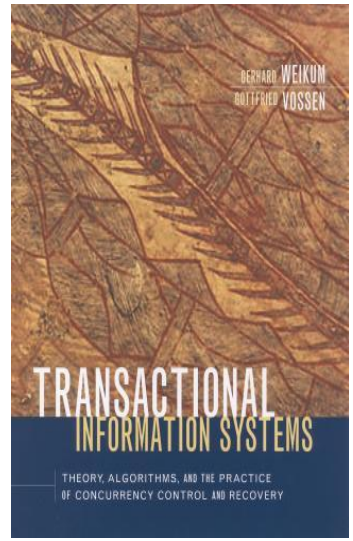
Transactional Information Systems:

Theory, Algorithms, and the Practice of Concurrency Control and Recovery

Gerhard Weikum and Gottfried Vossen

ISBN 1-55860-508-8



"Teamwork is essential. It allows you to blame someone else."(Anonymous)

Part III: Recovery

- 11 Transaction Recovery
- 12 Crash Recovery: Notion of Correctness
- 13 Page-Model Crash Recovery Algorithms
- 14 Object-Model Crash Recovery Algorithms
- 15 Special Issues of Recovery
- 16 Media Recovery
- 17 Application Recovery

Chapter 12: Crash Recovery – Notion of Correctness

- **12.2 System Architecture and Interfaces**
- 12.3 System Model
- 12.4 Correctness Criterion
- 12.5 Roadmap of Algorithms
- 12.6 Lessons Learned

"We will meet again if your memory serves you well. " (Bob Dylan)

Goal of Crash Recovery

Failure-resilience:

- **redo** recovery for committed transactions
- **undo** recovery for uncommitted transactions

Failure model:

- soft (no damage to secondary storage)
- fail-stop (no unbounded failure propagation)

captures most (server) software failures,
both Bohrbugs and Heisenbugs

Requirements:

- fast restart for high availability (= MTTF / (MTTF + MTTR))
- low overhead during normal operation
- simplicity, testability, very high confidence in correctness

Examples

- Server fails once a month, recovery takes 2 hours
  ⇒ 720/722 = 0,997
  i.e., server availability is 99,7 %
  server is down 26 hours per year

- Server fails every 48 hours, but can recover within 30 sec
  ⇒ 172800/172830 = 0,9998
  i.e., server availability is 99,98 %
  server is down 105 min per year

- Fast recovery is essential, not long uptime!

Actions During Normal Operation

All of the following actions are "tagged" with unique, monotonically increasing **sequence numbers**

Transaction actions:

- *begin (t)*
- *commit (t)*
- *rollback (t)*
- *save (t)*
- *restore (t, s)*

Data actions:

- *read (pageno, t)*
- *write (pageno, t)*
- *full-write (pageno, t)*
- *exec (op, obj, t)*

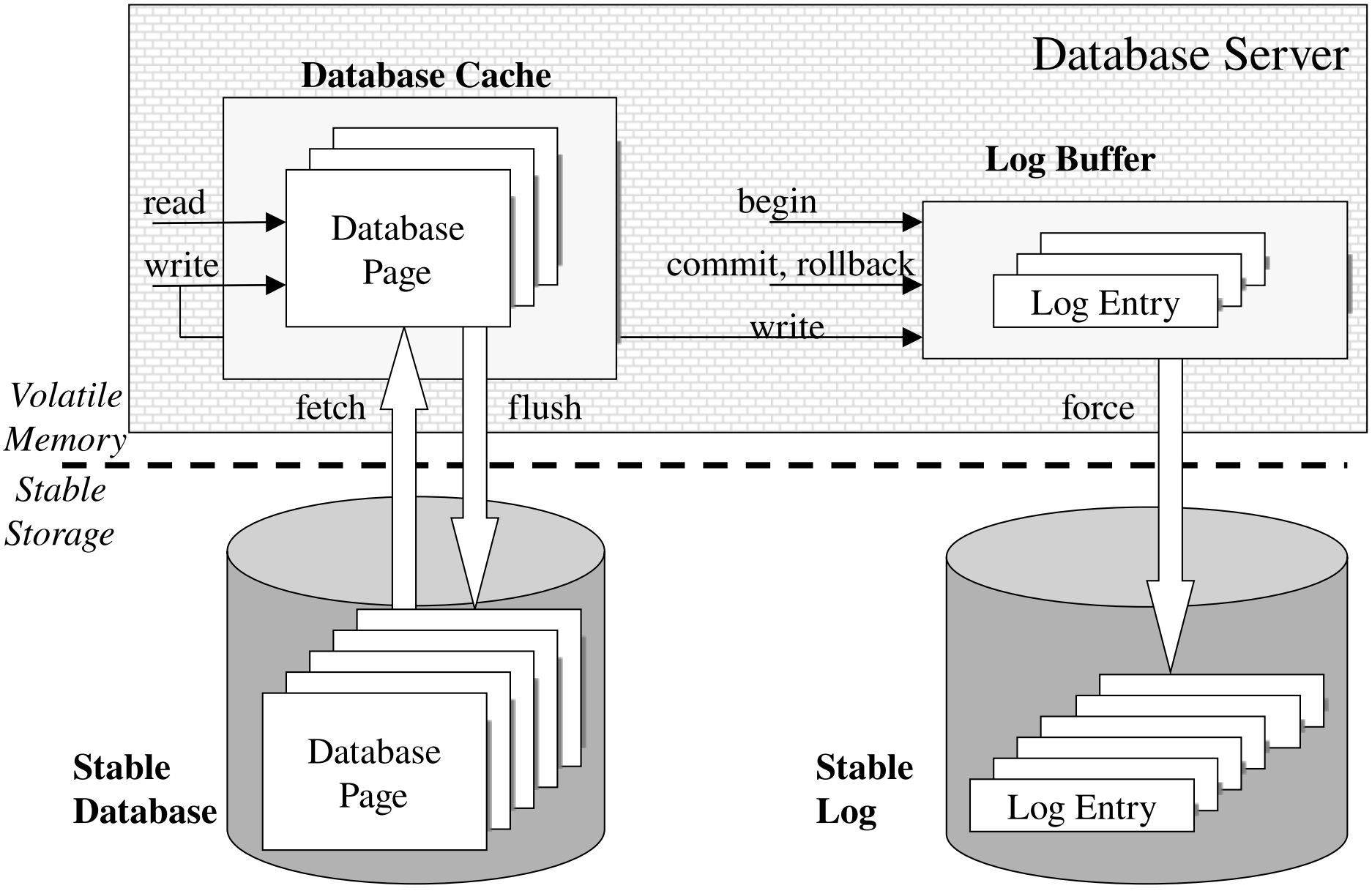
Caching actions:

- *fetch (pageno)*
- *flush (pageno)*

Log actions:

- *force ( )*

Overview of System Architecture



Chapter 12: Crash Recovery – Notion of Correctness

- 12.2 System Architecture and Interfaces
- **12.3 System Model**
- 12.4 Correctness Criterion
- 12.5 Roadmap of Algorithms
- 12.6 Lessons Learned

Logging

Definition 12.1 (Extended History):
The **extended history** of a transactional data server is a partially ordered forest of actions where

- the roots are transaction identifiers or caching actions,
- the leaves are read, write, or full-write actions or transaction actions,
- only exec actions can appear as intermediate nodes, and
- the ordering of actions is tree-consistent.

Definition 12.2 (Stable Log):
For a given extended history the **stable log** is a totally ordered subset of the history's actions such that the log ordering is compatible with the history order.

Definition 12.3 (Log Buffer):
For a given extended history the **log buffer** is a totally ordered subset of the history's actions such that the log ordering is compatible with the history order and all entries in the log buffer follow (w.r.t. the total order) all entries in the stable log.

Impact of Caching

Definition 12.4 (Cached Database):
For a given extended history the **cached database** is a partially ordered subset of the history's write actions such that the order is a subset of the the history order, and for each page p the maximum element among the write actions on p in the history is also the maximum element for p in the cached database.

Definition 12.5 (Stable database):
For a given extended history the **stable database** is a partially ordered subset of the history's write actions such that the order is a subset of the history order, and for each page p

- all write actions on p that precede the most recent flush(p) in the history are included in the stable database, and
- the maximum element among all included write actions in the history is also the maximum element for p in the stable database.

The maximum element among all writes on a page p is tracked by the **page sequence number** in the header of p.

Chapter 12: Crash Recovery – Notion of Correctness

- 12.2 System Architecture and Interfaces
- 12.3 System Model
- **12.4 Correctness Criterion**
- 12.5 Roadmap of Algorithms
- 12.6 Lessons Learned

Correctness Criterion

Definition 12.6 (Correct Crash Recovery):
A crash recovery algorithm is **correct** if it guarantees that,
after a system failure, the cached database will eventually,
i.e., possibly after repeated failures and restarts,
be equivalent (i.e., reducible) to
a serial order of the committed transactions
that coincides with the serialization order of the history.

Logging Rules

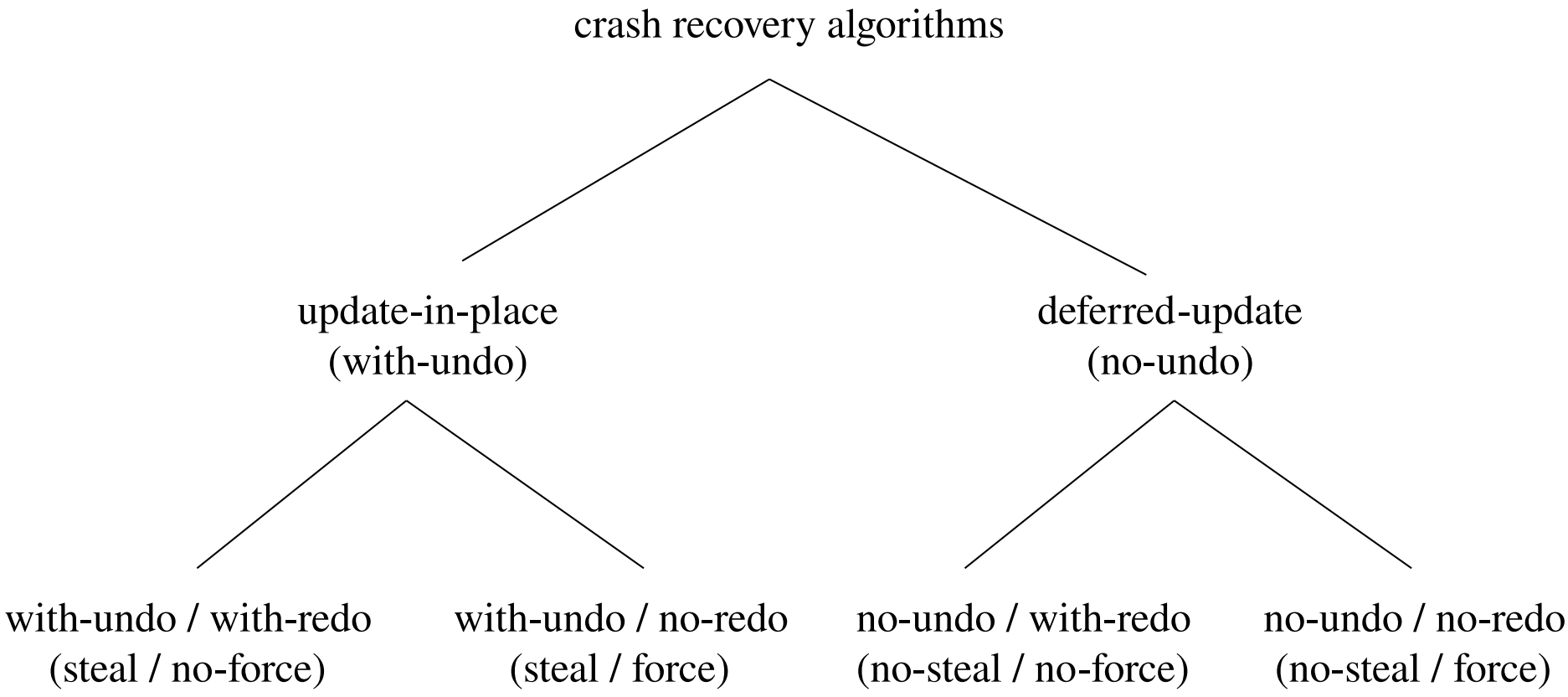
Definition 12.7 (Logging Rules):
During normal operation, a recovery algorithm satisfies

- the **redo logging rule** if for every committed transaction t, all data actions of t are in the stable log or the stable database,
- the **undo logging rule** if for every data action p of an uncommitted transaction t the presence of p in the stable database implies that p is in the stable log,
- the **garbage collection rule** if for every data action p of transaction t the absence of p from the stable log implies that p is in the stable database if and only if t is committed.

Chapter 12: Crash Recovery – Notion of Correctness

- 12.2 System Architecture and Interfaces
- 12.3 System Model
- 12.4 Correctness Criterion
- **12.5 Roadmap of Algorithms**
- 12.6 Lessons Learned

Taxonomy of Crash-Recovery Algorithms



steal/no-force algorithms are most versatile and cost-effective

Chapter 12: Crash Recovery – Notion of Correctness

- 12.2 System Architecture and Interfaces
- 12.3 System Model
- 12.4 Correctness Criterion
- 12.5 Roadmap of Algorithms
- **12.6 Lessons Learned**

Lessons Learned

- During normal operation and during restart,
  operations are captured in the log buffer, the stable log,
  the cached database, and the stable database.
- Correct recovery requires preserving the original serialization
  order of the committed transactions.
- The redo logging, undo logging, and garbage collection rules
  are necessary prerequisites for the ability to provide correct recovery.