Decoupled Propagation for DBMS Architectures

Lucas Lersch

M. Sc. Caetano Sauer
Advisor

21.12.2015
Motivation
Motivation

- Today’s databases:
  - large amounts of memory → large portion of working set in buffer pool
  - up-to-date version of database in persistent storage?
    - RECOVERY!
- Propagation Services
  - checkpoints
  - page cleaner
- Architectural problem
Contribution

In-memory processing
(Data structures + buffer pool)

Tightly coupled

Persistence
(Recovery)

LOG

Page fetch

In-memory processing
(Data structures + buffer pool)

LOG
System Recovery

- **System crash**

  LOG: ...

- **Log analysis**

  LOG: ____________ forward scan

  dirty-pages list & active-transactions list

- **REDO (replay log-records)**

  LOG: ____________ forward scan

- **UNDO**

  LOG: ____________ backward scan

  compensation log-records
Checkpoints

- Checkpoints
  - reduce length of log analysis
  - no page flushing

LOG: ...

- Taking a checkpoint:

LOG: ...

Buffer Pool

Tx Manager

Lock Manager

BEGIN_CHKPT

dirty-pages list

active-transactions list

acquired-locks list

END_CHKPT

forward scan

dirty-pages list & active-transactions list
Decoupled Checkpoints

- Classical checkpoint:

- Decoupled checkpoint:
Decoupled Checkpoints

- Taking a decoupled checkpoint:

Buffer Pool  |  Tx Manager  |  Lock Manager

LOG:  OLD_CHKPT  |  BEGIN_CHKPT  |  dirty-pages list  |  active-transactions list  |  acquired-locks list  |  END_CHKPT

forward scan
Decoupled Checkpoints

- Decoupled-checkpoint considerations:
  - same algorithm of log analysis
  - no interference in in-memory data structures
  - requires I/O to forward-scan log records
  - requires page_write log records
Page Cleaner

- Page Flushing
  - page eviction
  - system shutdown
  - reduce REDO work

- Page Cleaner Service

![Diagram showing the interaction between Buffer Pool, Page Cleaner, and Database in the context of page cleaning and flushing.]

Buffer Pool

- Page Cleaner
  - candidates
  - sorted candidates
  - flush buffer

Database

FLUSH
Decoupled Page Cleaner

- Decoupled Page Cleaner
  - partially-sorted log archive*

---

*Caetano Sauer, Goetz Graefe and Theo Härder
"Single-pass restore after a media failure"
Decoupled Page Cleaner

- Decoupled Page Cleaner
  - partially-sorted log archive
  - indexed access to log records by page
Decoupled Page Cleaner

- Single-Page Recovery*

1. Is page up-to-date?
   - Yes → Good! :)
   - No → Apply SPR

*Goetz Graefe and Harumi A. Kuno
“Definition, detection, and recovery of single-page failures, a fourth class of database failures”
Decoupled Page Cleaner

- Write elision
  - evict dirty pages without flushing
Decoupled Page Cleaner

- Single-Page Recovery: Read elision
  - page is not fetched from persistent storage
  - update is merely logged, and applied later

- Both, write and read elision reduce system I/O costs

- Recovery without UNDO*
  - database always in a committed state
    - no dirty updates

- Different page format
  - in-memory page
  - persistent page

*Caetano Sauer and Theo Harder
“A novel recovery mechanism enabling fine-granularity locking and fast, redo-only recovery.”
Experiments

- Zero storage manager* (based on Shore-MT)
- TPC-B (large amount of small read-write transactions)
  - 15 minutes
- SF == #Threads (24)
  - no concurrent transaction conflicts
- Buffer pool size → 100% (~5GB)
- Database & log archive
  - SSD
- Recovery log
  - SSD
  - Memory

* https://github.com/caetanosauer/zero
Checkpoint Results

Log in SSD:

Log in MEM:
Checkpoint Results

Log in SSD:

Log in MEM:
Cleaner Results
Cleaner Results

Log in SSD:

Log in MEM:
Future Work

- Experiments in a more realistic benchmark environment
- Profiler analysis
  - isolate components
- In-depth analysis of cleaner behavior and page access patterns
- Write elision
Conclusion

- Decoupled architecture
  - reduce interference in in-memory data structures
  - modular design
  - less code complexity
    - eliminates interaction between components
    - easier concurrent programming
Conclusion

Thank you!