TECHNISCHE UNIVERSITÄT
KAISERSLAUTERN

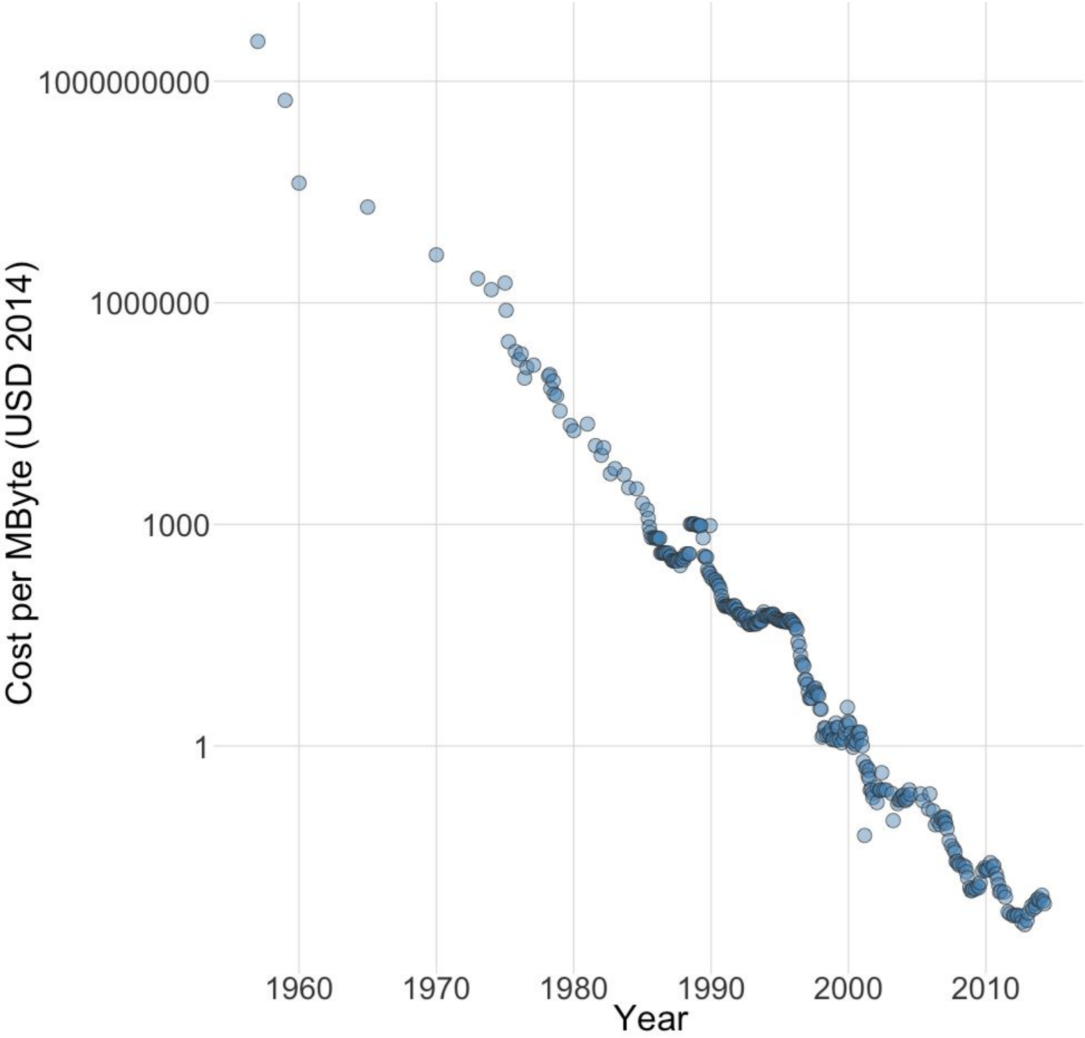
# Decoupled Propagation for DBMS Architectures

Lucas Lersch

M. Sc. Caetano Sauer
Advisor

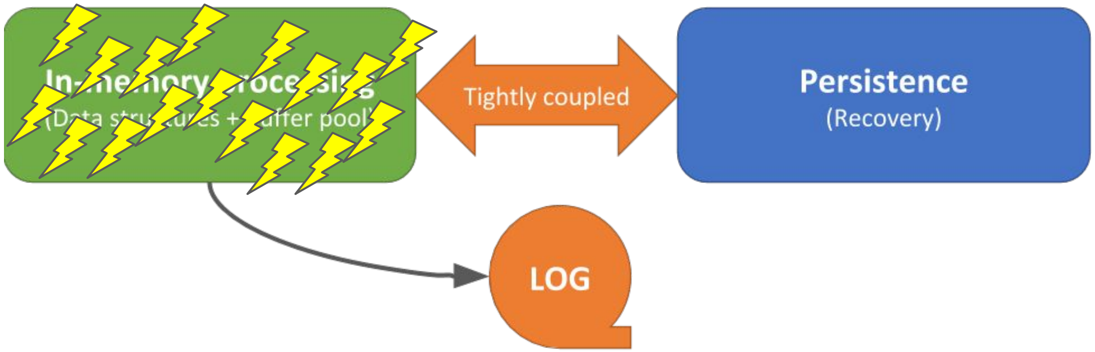
21.12.2015

# Motivation

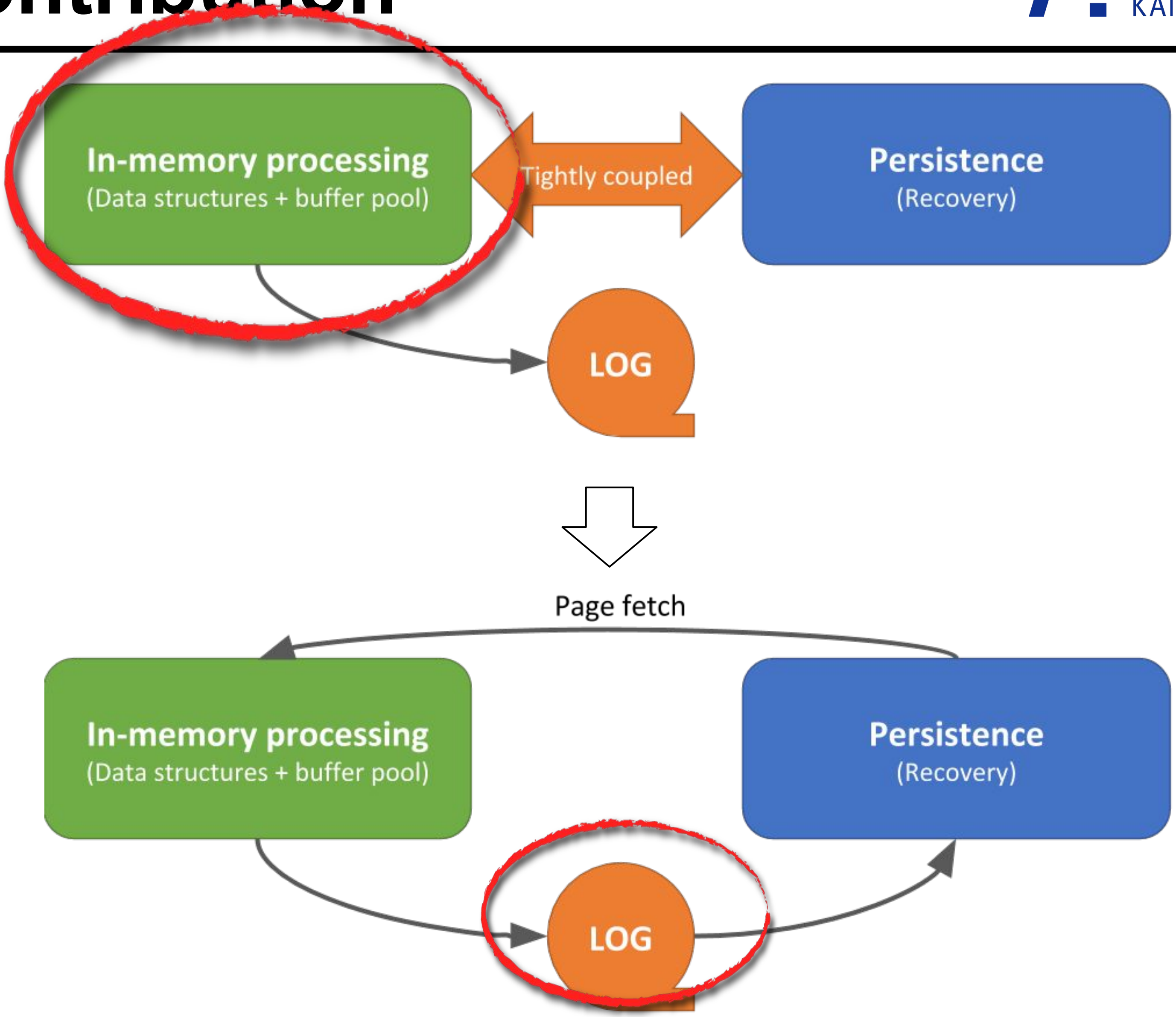

1000000000
1000000
1000
1
Cost per MByte (USD 2014)
1960
1970
1980
1990
2000
2010
Year

# 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

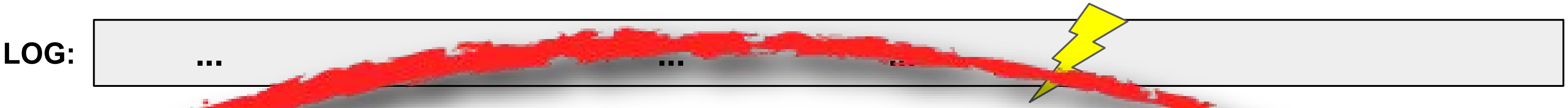


# Contribution

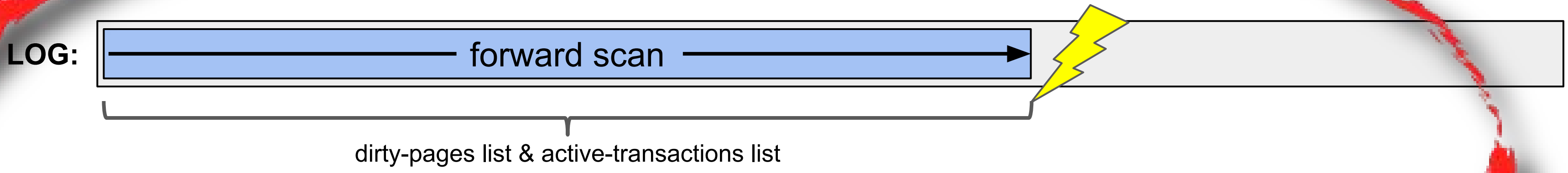


# System Recovery

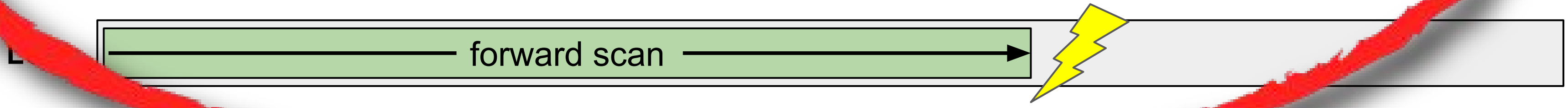
- System crash



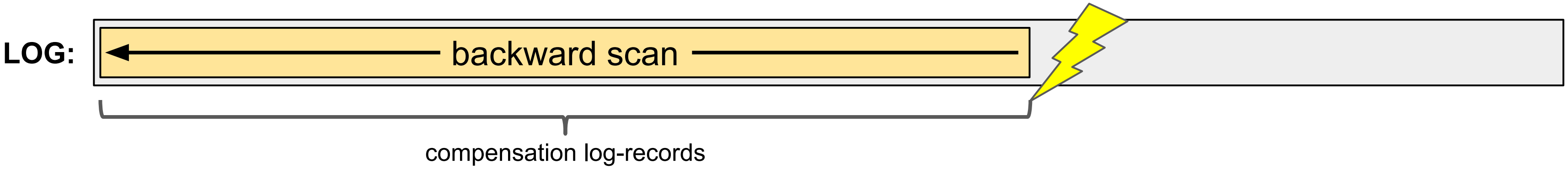
- Log analysis



- REDO (replay log-records)

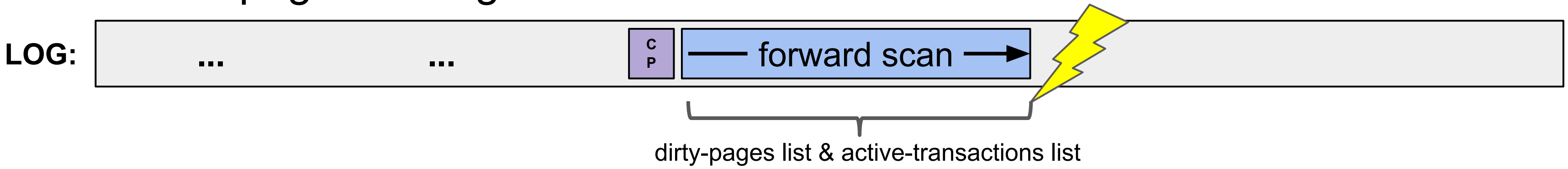


- UNDO

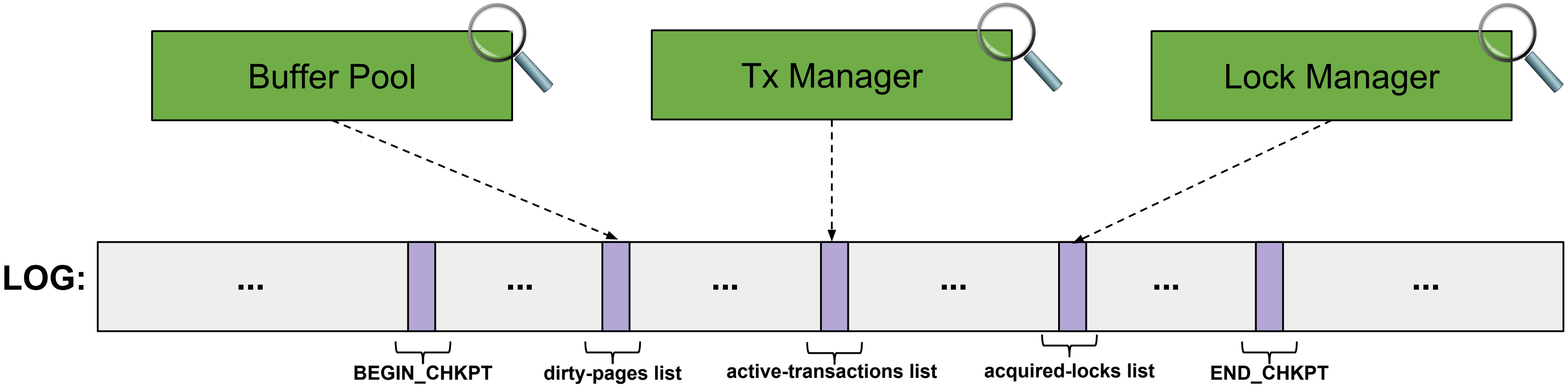


# Checkpoints

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

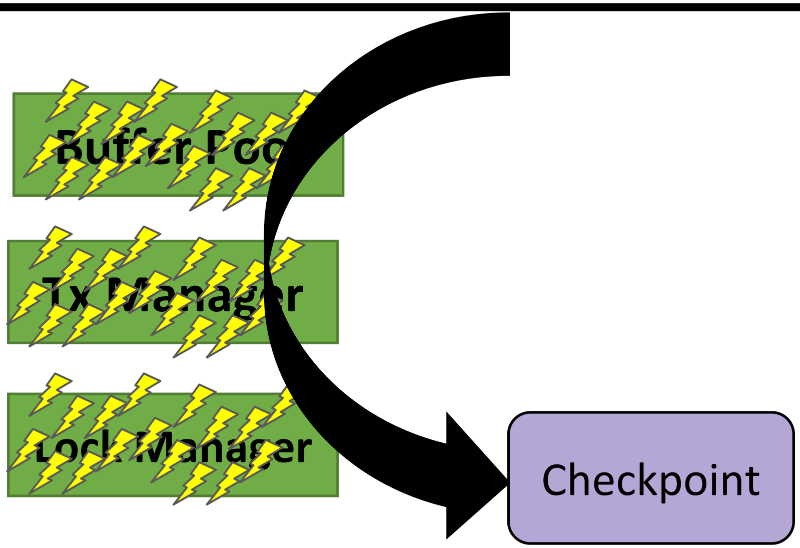


- Taking a checkpoint:

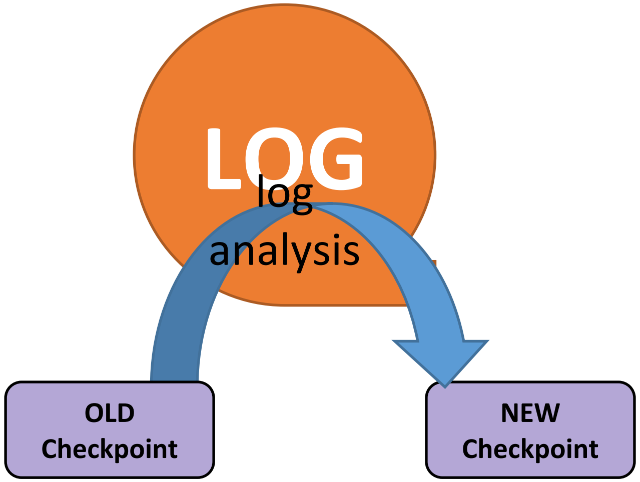


# Decoupled Checkpoints

- Classical checkpoint:

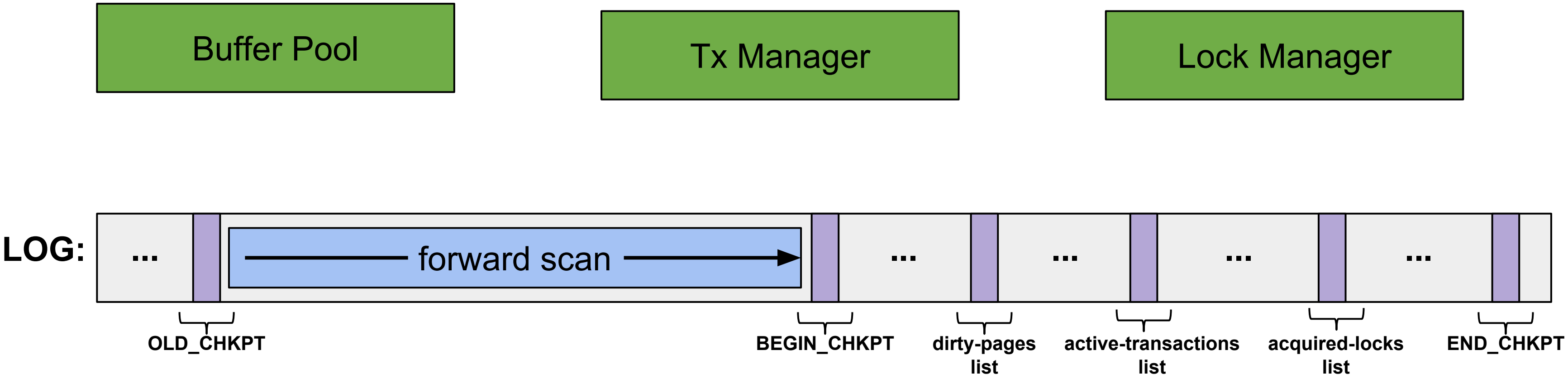


- Decoupled checkpoint:



# Decoupled Checkpoints

- Taking a decoupled checkpoint:



# Decoupled Checkpoints

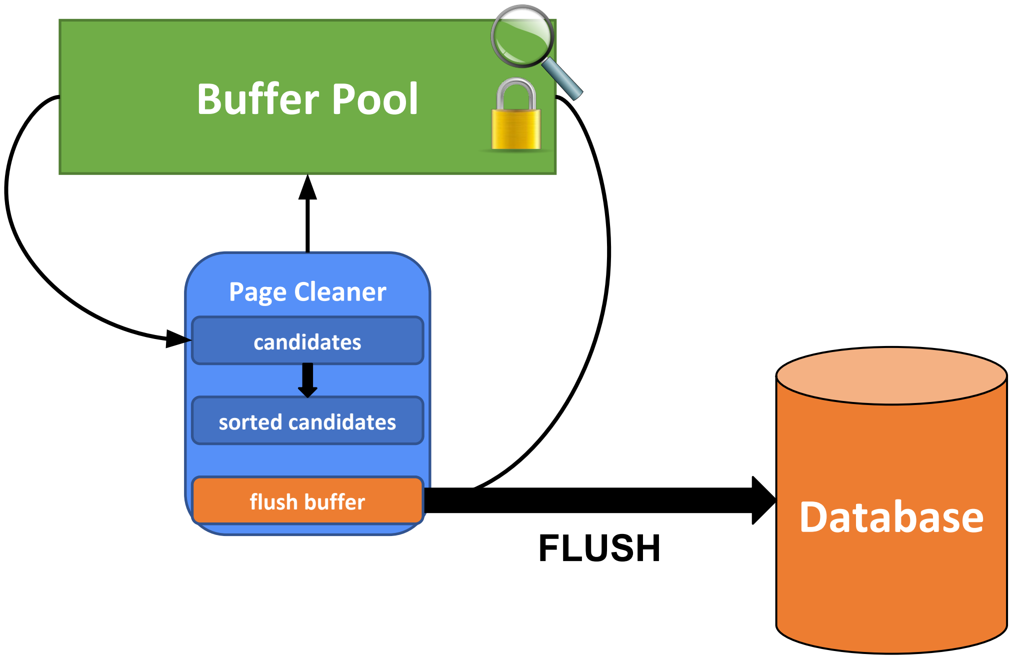
- Decoupled-checkpoint considerations:
  - same algorithm of log analysis
  - no interference in in-memory data structures
  - requires I/O to forward-scan log records
  - requires `page_write` log records

# Page Cleaner

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

  } Page Cleaner Service

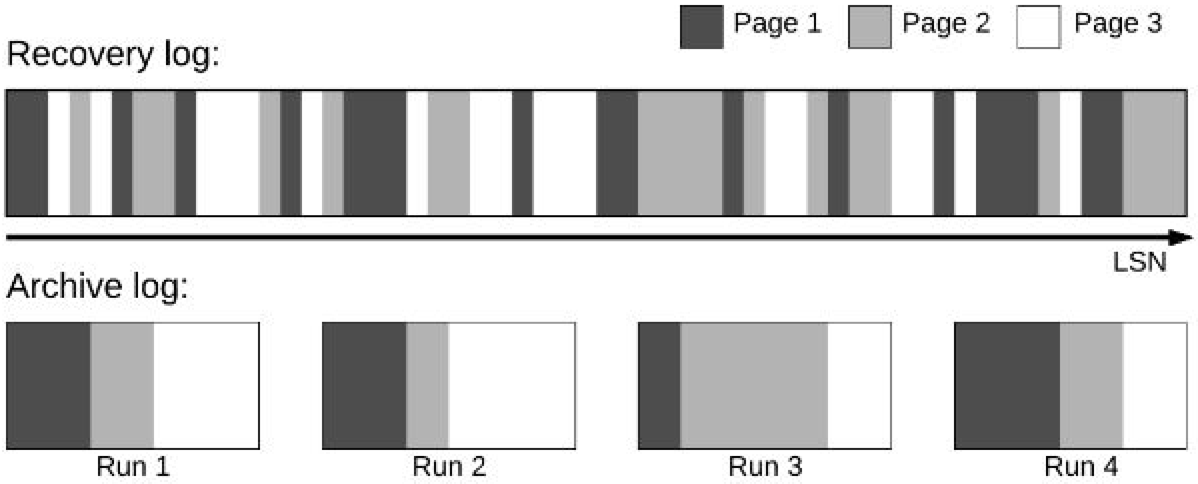
- Page Cleaner Service



# Decoupled Page Cleaner

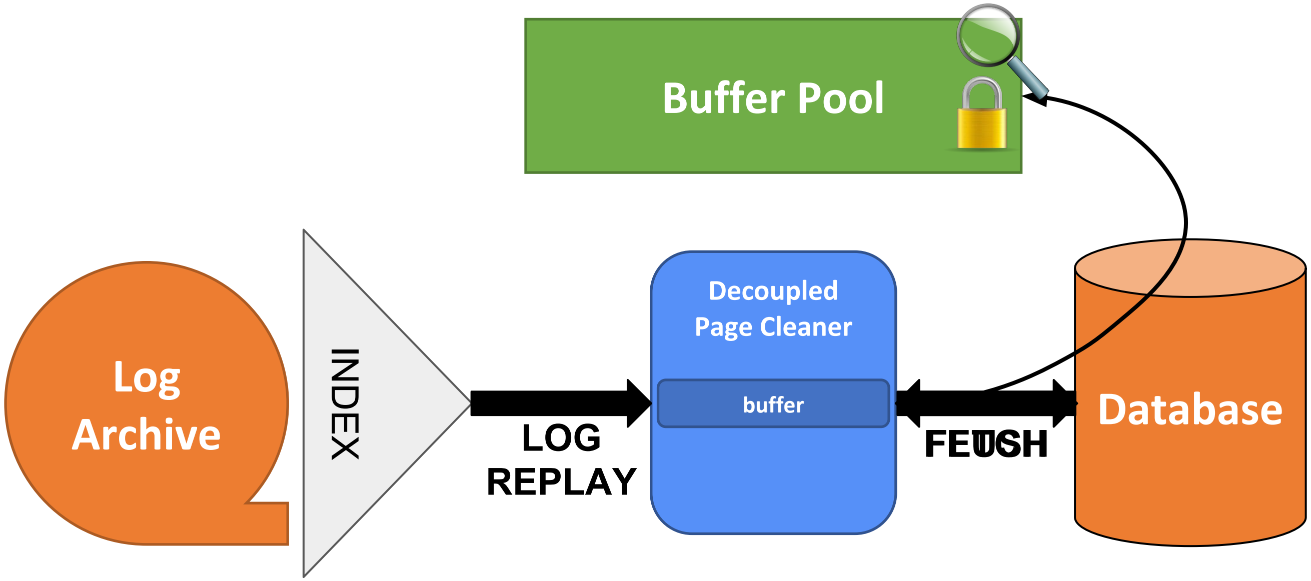
*Caetano Sauer, Goetz Graefe and Theo Härder
*"Single-pass restore after a media failure"*

- Decoupled Page Cleaner
  - partially-sorted log archive*



# Decoupled Page Cleaner

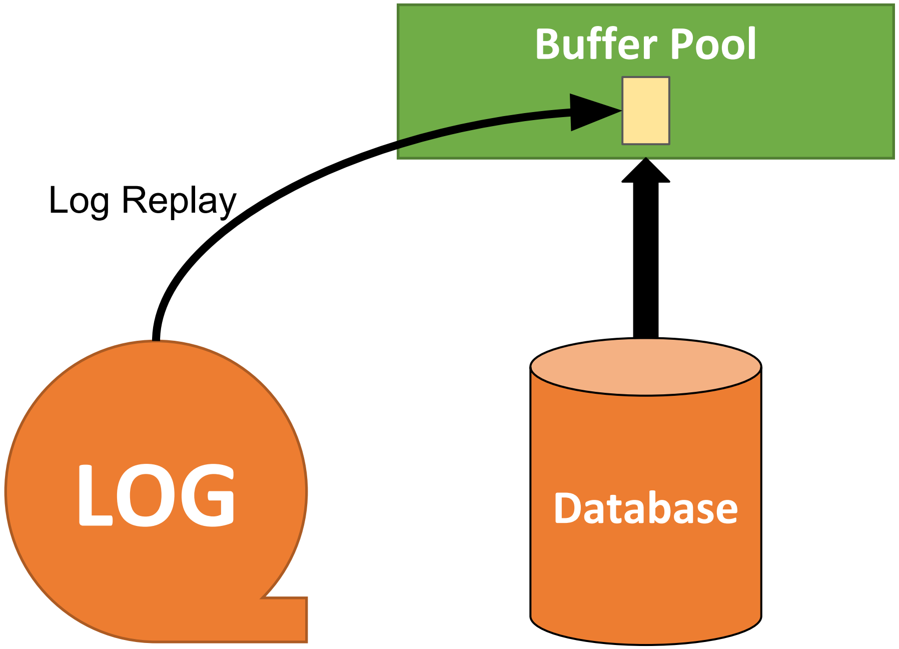
- Decoupled Page Cleaner
  - partially-sorted log archive
    - indexed access to log records by page



# Decoupled Page Cleaner

- Single-Page Recovery*

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



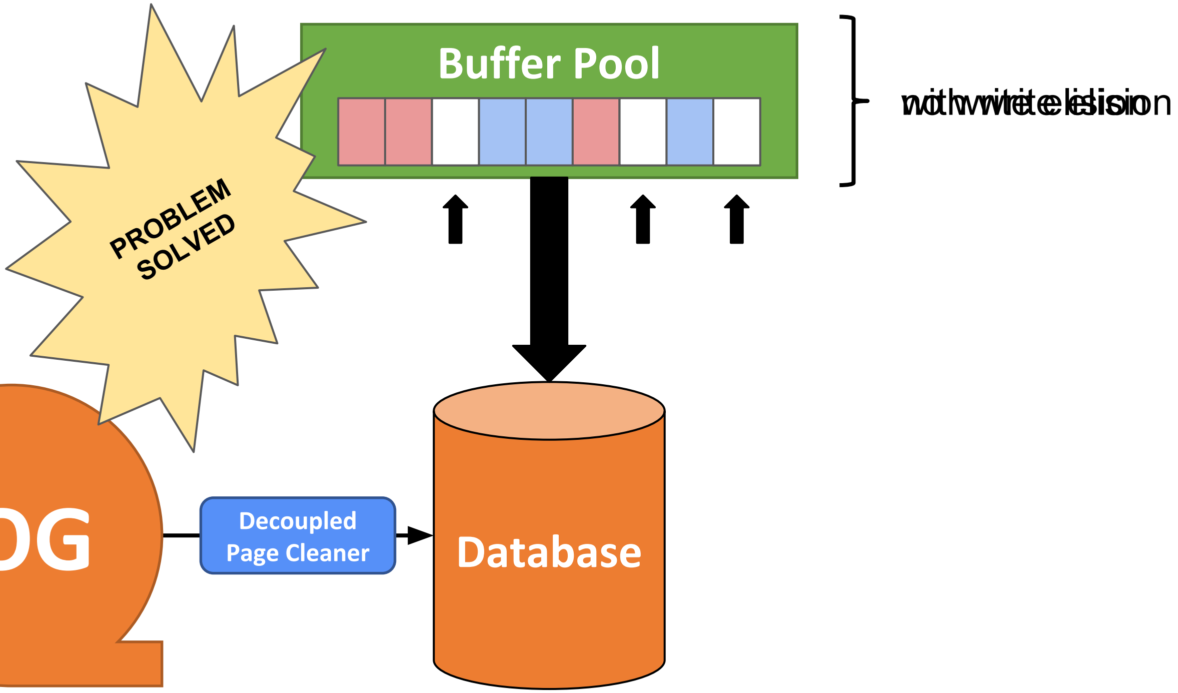
1. Is page up-to-date?

Yes → Good! :)

No → Apply SPR

# 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."

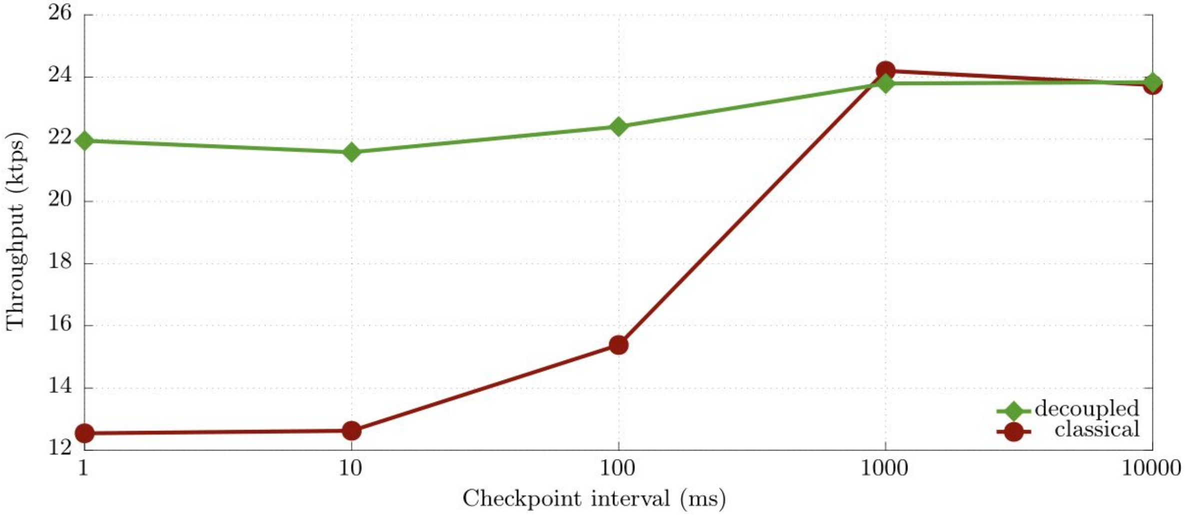
# Experiments

- 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

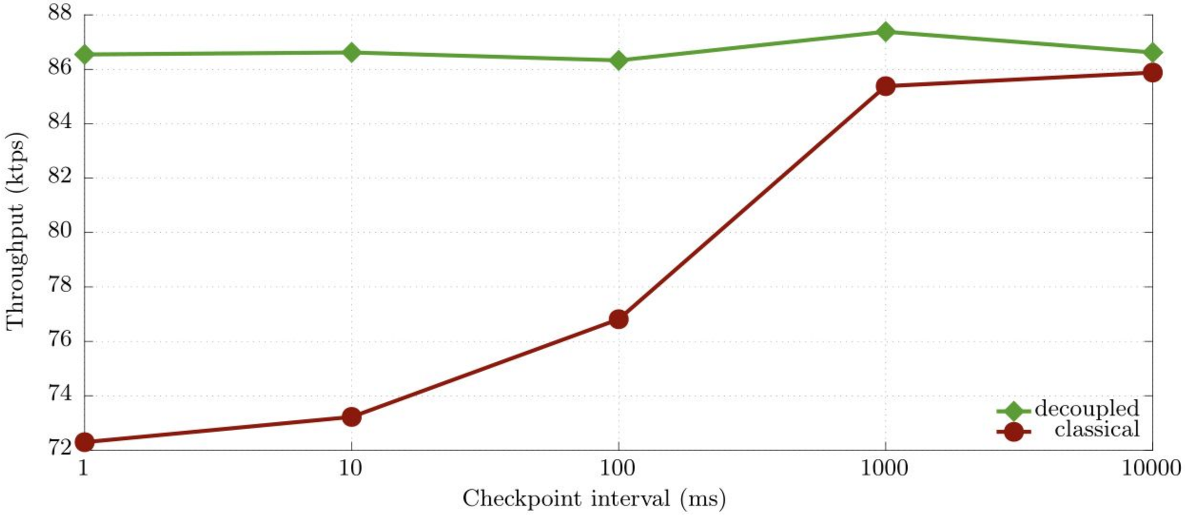
[*] https://github.com/caetanosauer/zero

# Checkpoint Results

Log in SSD:

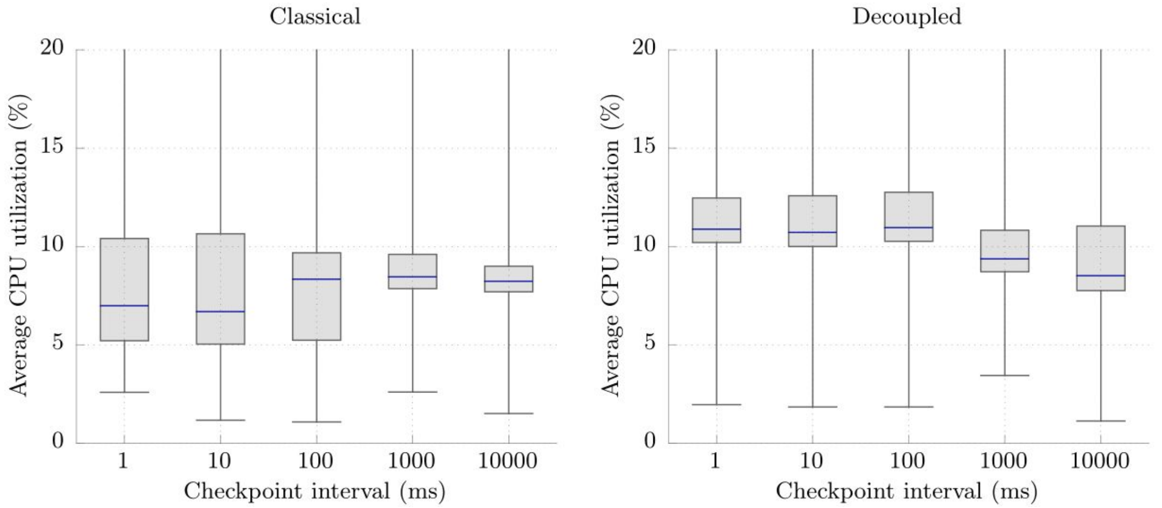


Log in MEM:

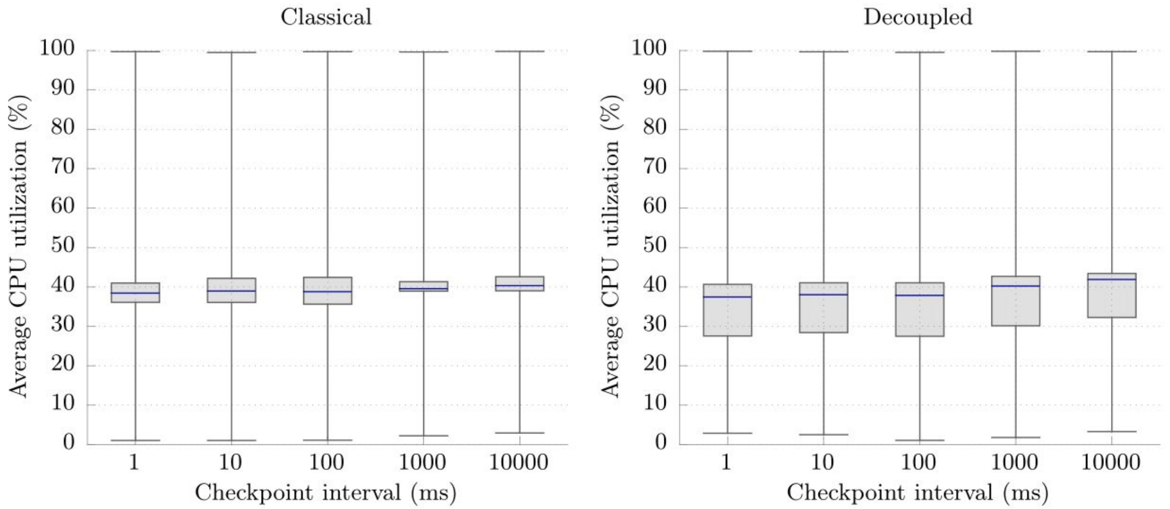


# Checkpoint Results

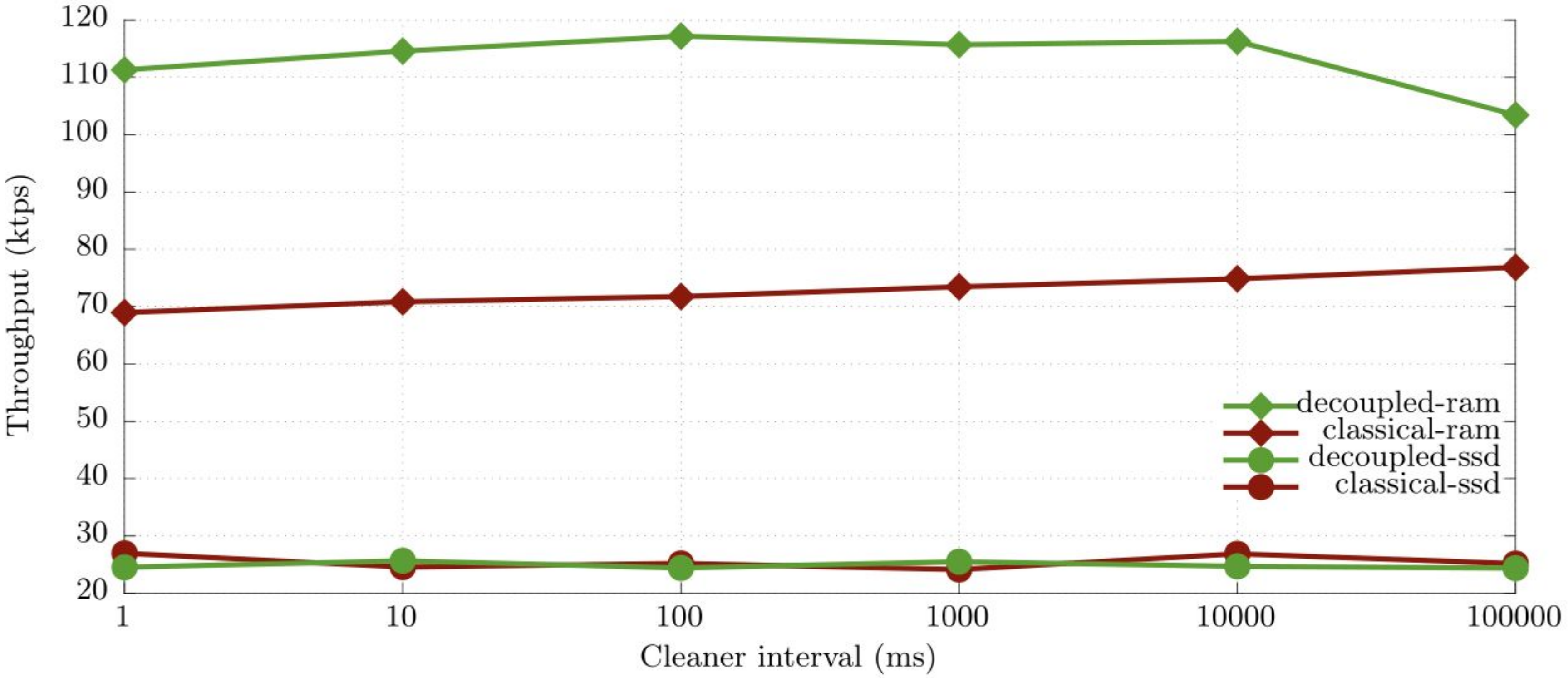
**Log in SSD:**



**Log in MEM:**

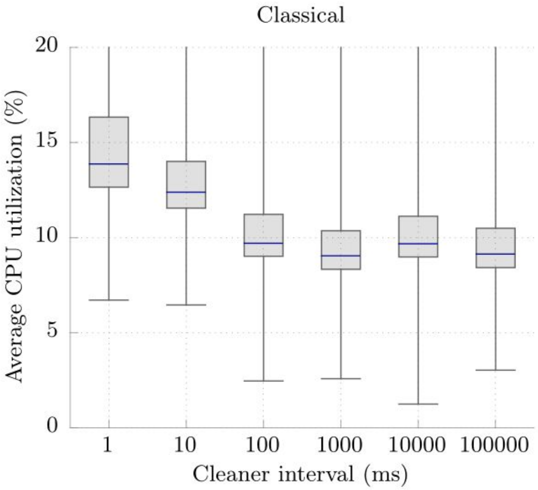


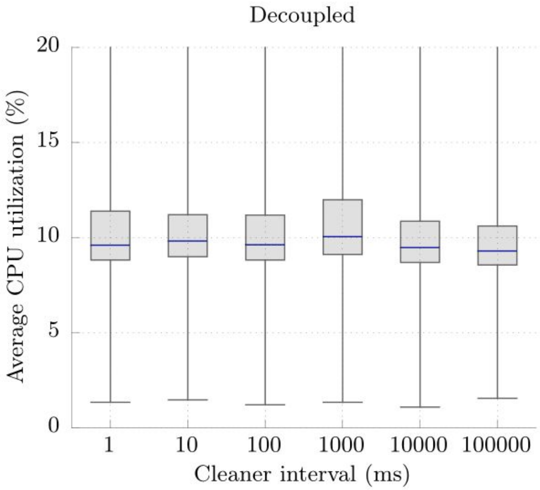
# Cleaner Results



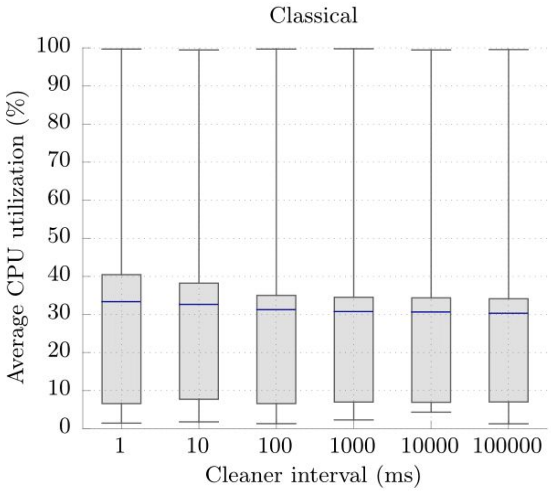
# Cleaner Results

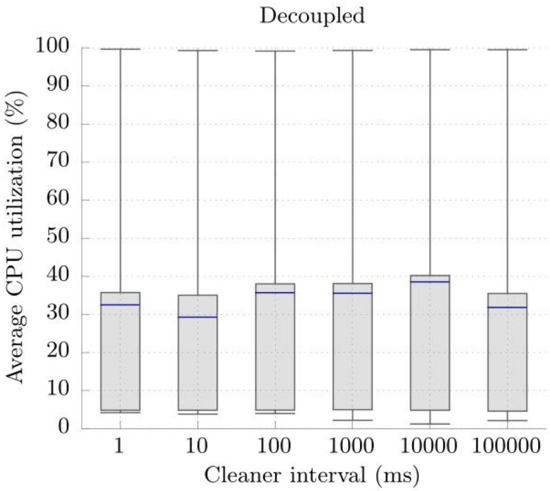
**Log in SSD:**





**Log in MEM:**





# Future Work

- Experiments in a more realistic benchmark environment

- Profiler analysis
  - isolate components

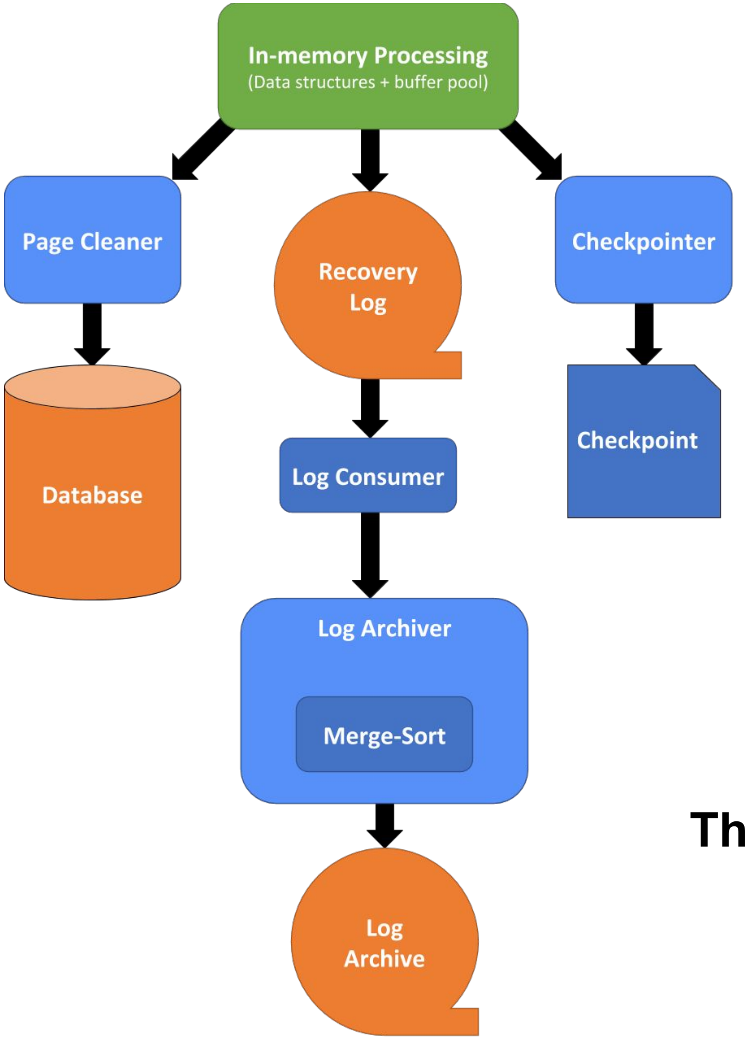
- In-depth analysis of cleaner behavior and page access patterns

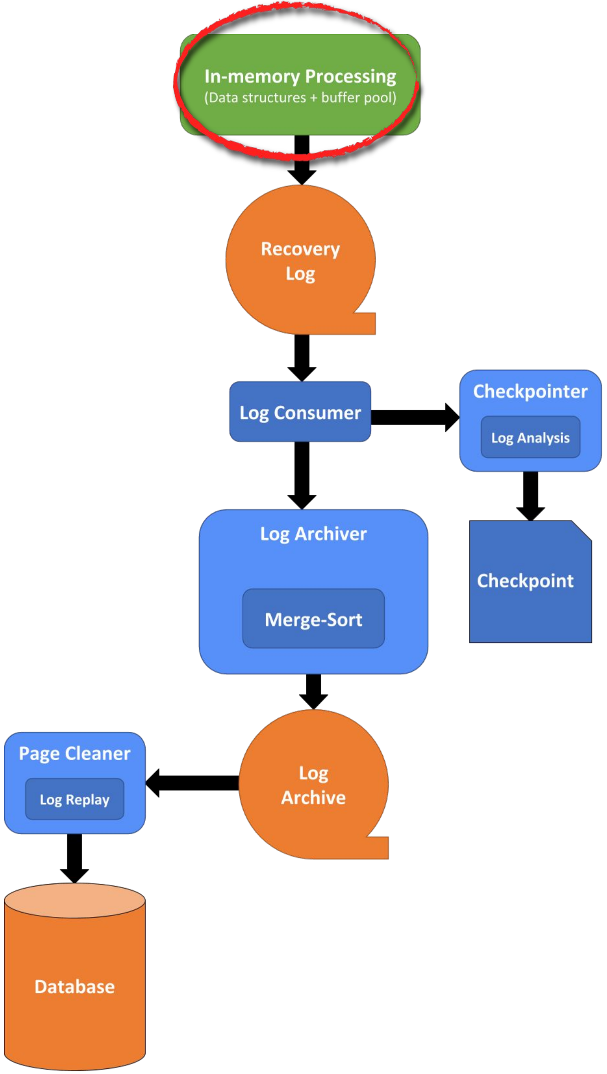
- Write elision

# 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!**