Fast Databases with Fast Durability and Recovery Through Multicore Parallelism

Wenting Zheng, Stephen Tu (MIT/UC Berkeley)
Eddie Kohler (Harvard), Barbara Liskov (MIT)

Motivation

- In-memory databases are popular
  - extremely fast transaction processing
  - VoltDB, MemSQL, etc.

Motivation

- In-memory databases are popular
  - extremely fast transaction processing
  - VoltDB, MemSQL, etc.

Potential weakness: durability!

Need a persistence system with

small performance impact on runtime throughput and latency

and

recovery of a big database in a few minutes

for a fast, multicore, in-memory database

Challenges

- Avoid interference with transaction execution

Challenges

- Avoid interference with transaction execution
- Fast recovery
  - serial recovery takes too long
  - parallel recovery constrains logging and checkpointing designs

Solution

- SiloR provides persistence for Silo (SOSP '13)
  - logging, checkpointing, recovery using disks

Solution

- SiloR provides persistence for Silo (SOSP '13)
  - logging, checkpointing, recovery using disks

IDEA: **parallelism** in all parts of the system, both runtime and recovery

- **Silo Overview**
- SiloR Design
  - Logging
  - Checkpointing
  - Recovery
- Evaluation
- Related work

Silo Overview

- Silo is a very high performance in-memory database
- Workers on different cores execute transactions on a shared-memory database
- Optimistic concurrency control (OCC)

Silo TID and Epochs

- Epochs are global time periods (~40 ms)
- Silo TIDs are grouped into epochs
- Writes ordered by TIDs
- Epochs provide group commit and avoid contention on global TID
- Epochs are recovery units

- Silo Overview
- **SiloR Design**
  - **Logging**
  - Checkpointing
  - Recovery
- Evaluation
- Related work

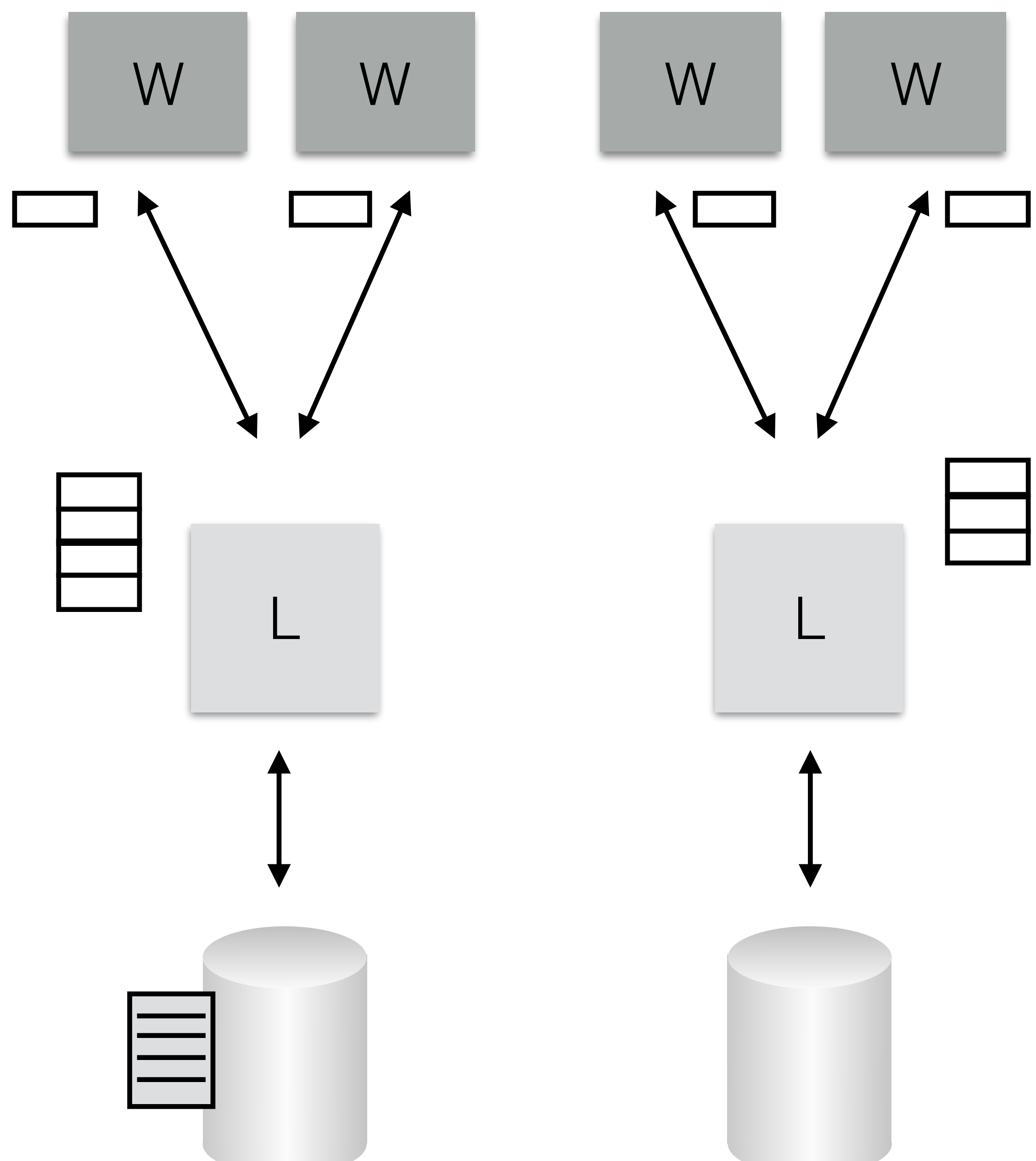
Logging

- Operation vs. value logging
  - Operation logging: smaller log size
  - Value logging: easier to parallelize recovery
- SiloR uses *value logging*

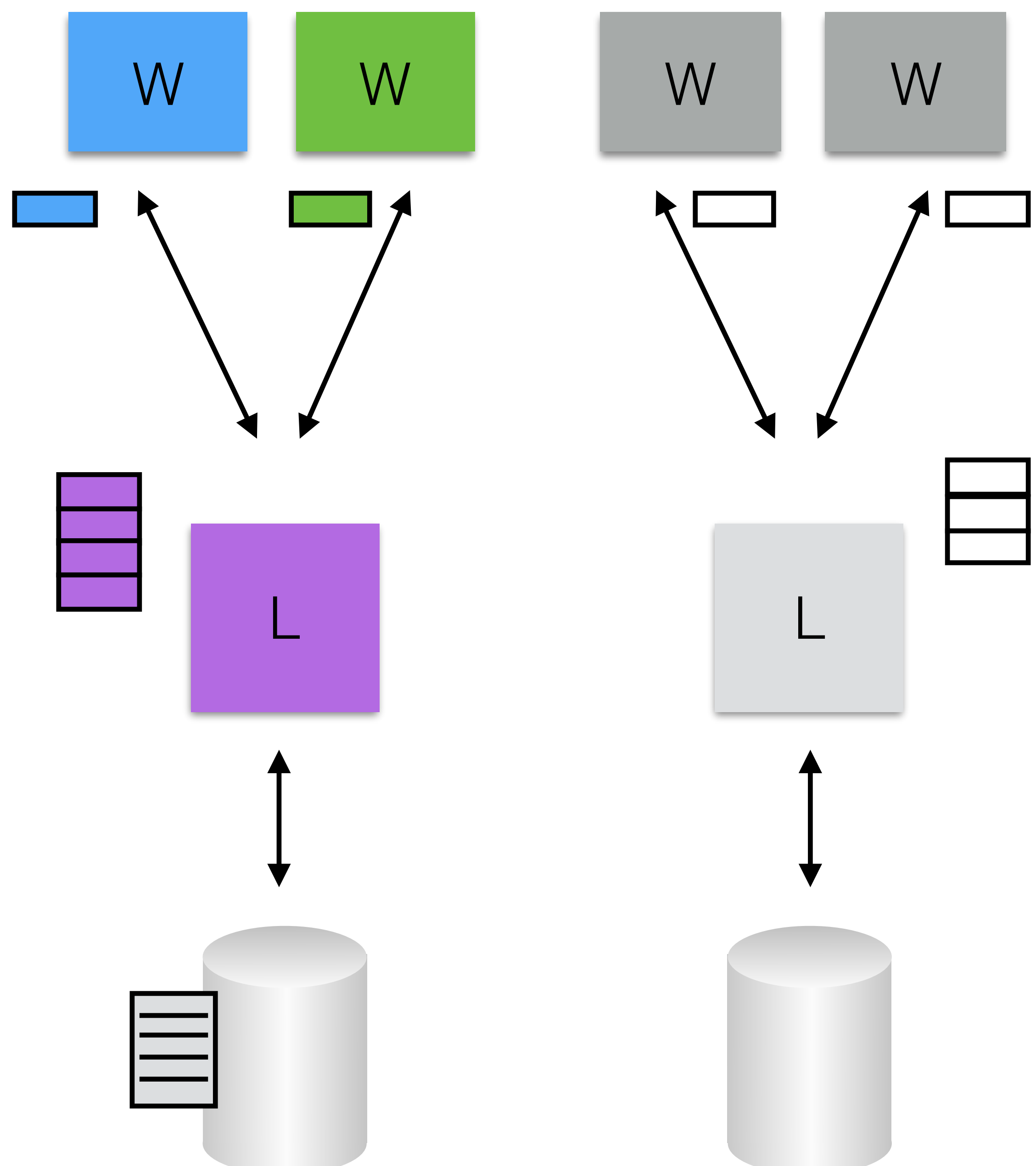
Logging Parallelism

- Must use multiple disks - single disk's IO not enough
- One logger per disk
- Multiple workers for one logger

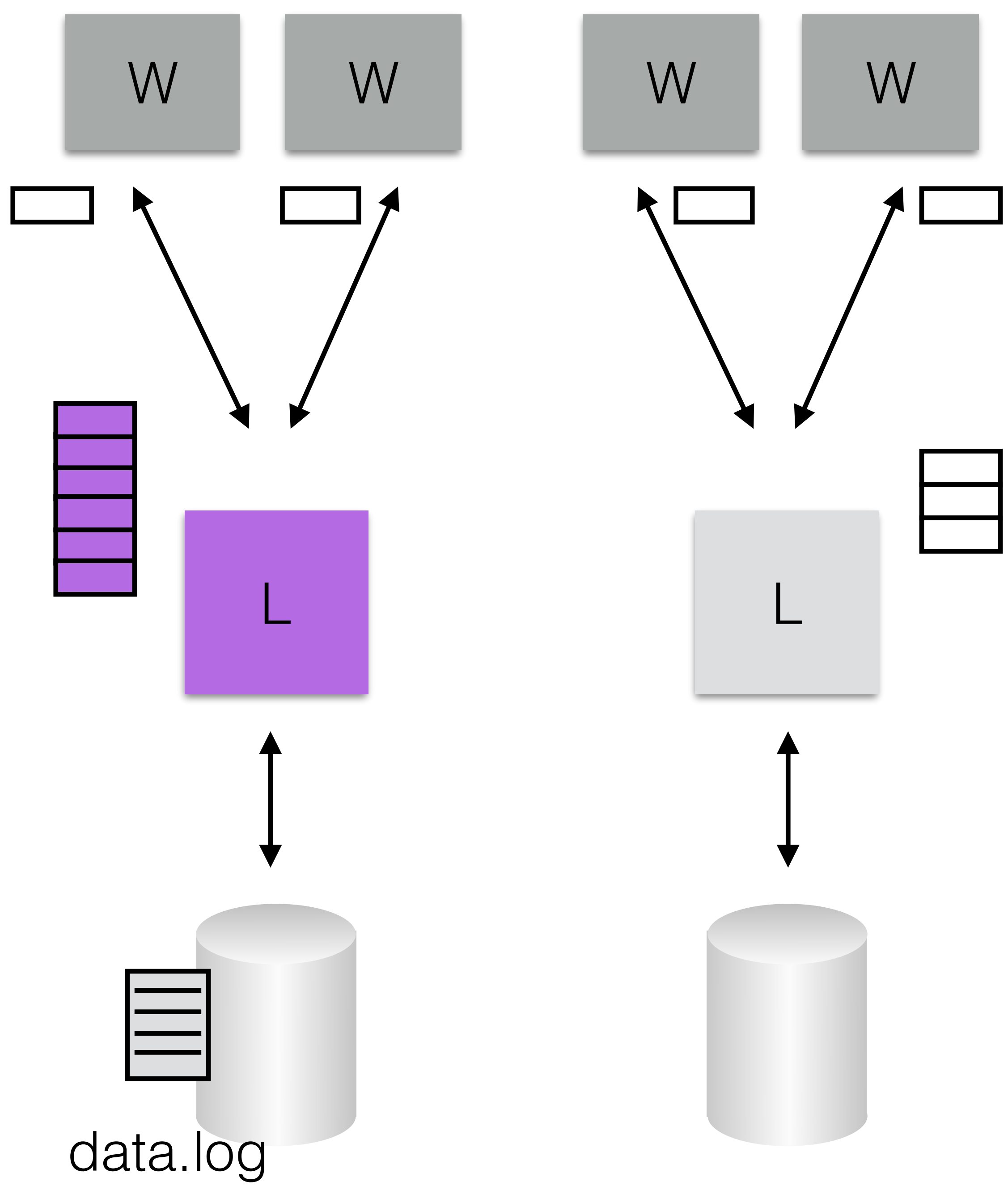
Logging structure



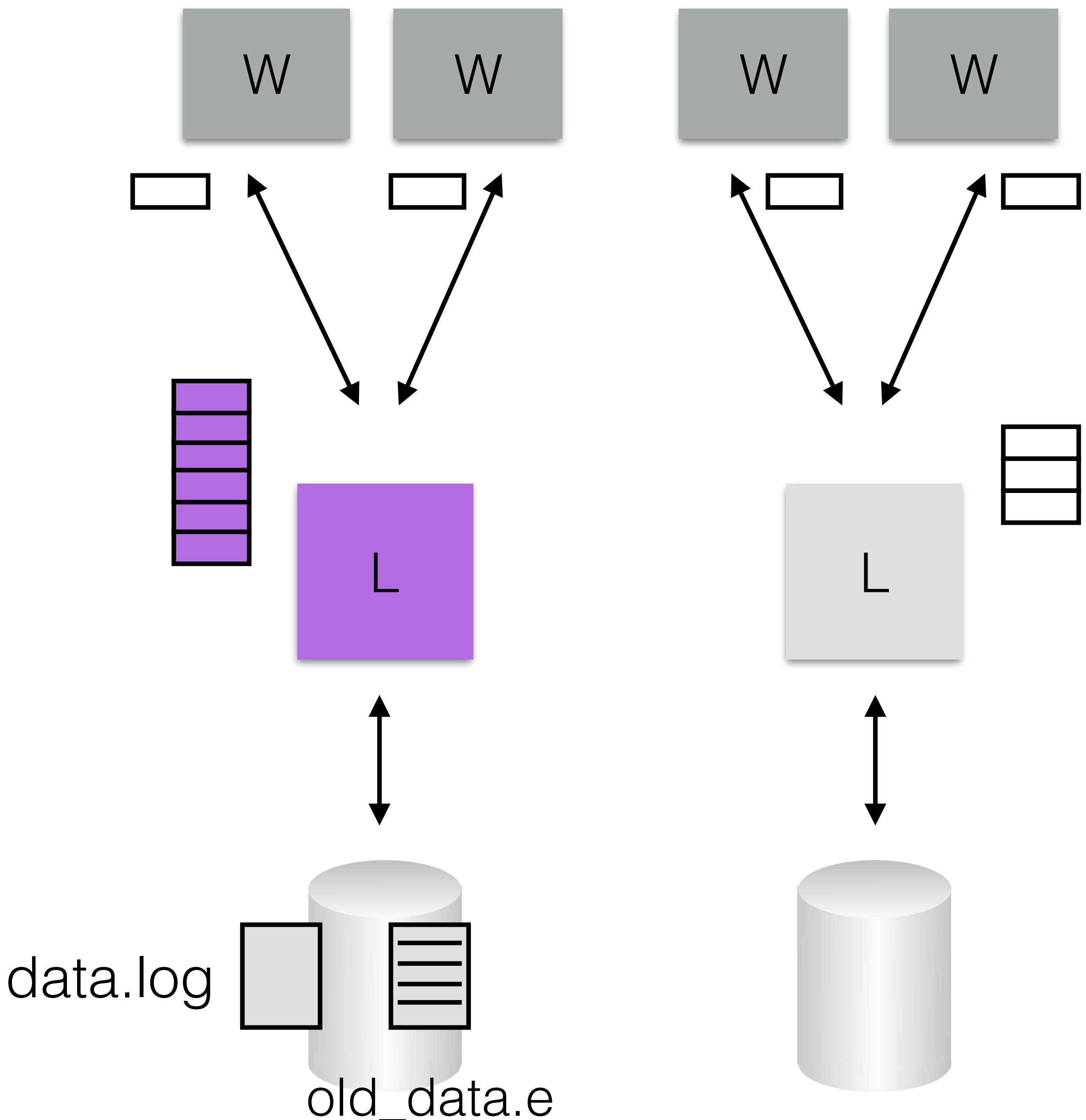
Logging structure



Log rotation

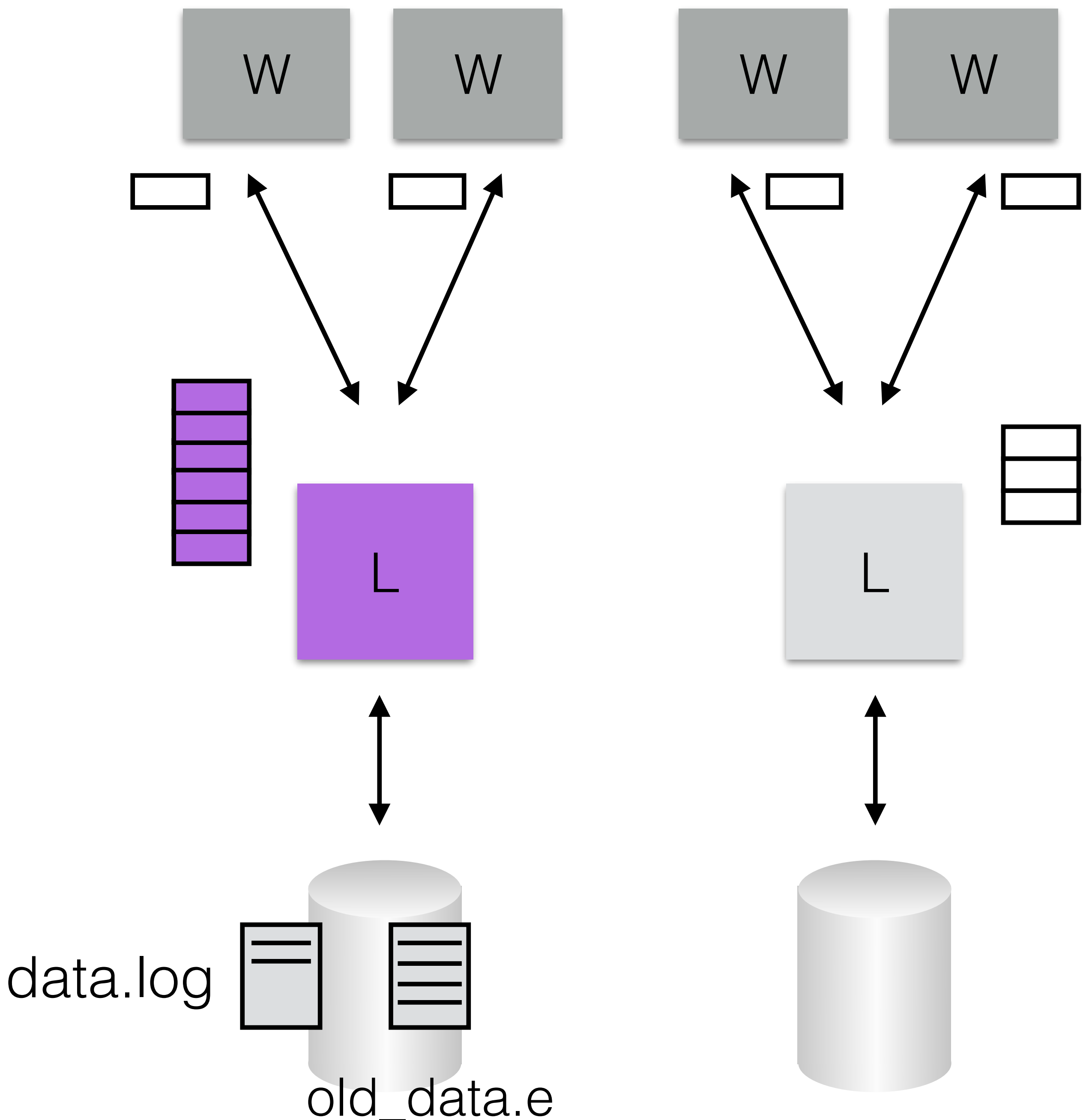


Log rotation



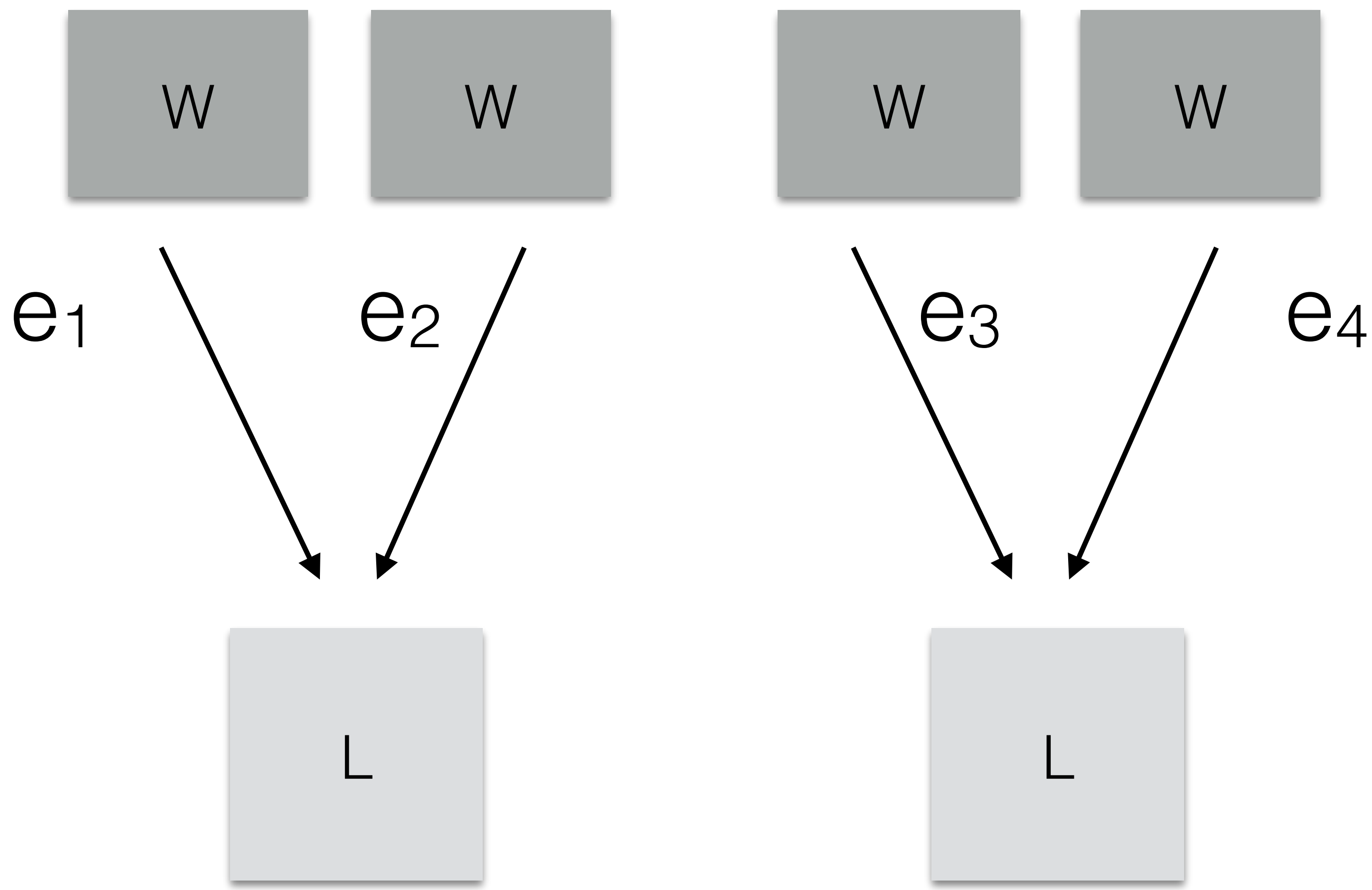
Log file renamed to **old_data.e**, where **e** is the largest epoch seen in that particular file.

Log rotation



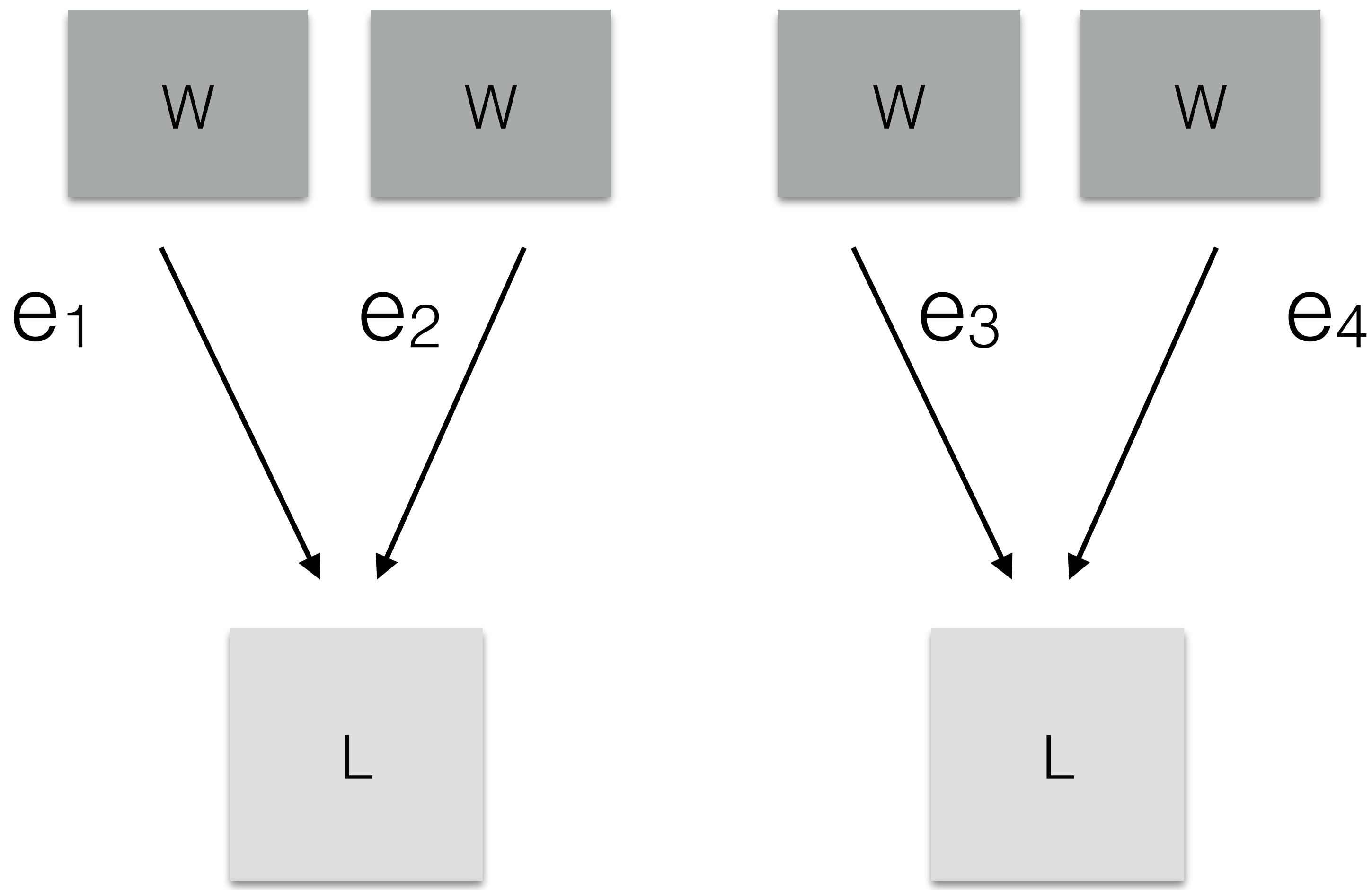
Log file renamed to **old_data.e**, where **e** is the largest epoch seen in that particular file.

Persistence epoch



$$\mathbf{e_P} = \min\{e_1, e_2, e_3, e_4\} - 1$$

Persistence epoch

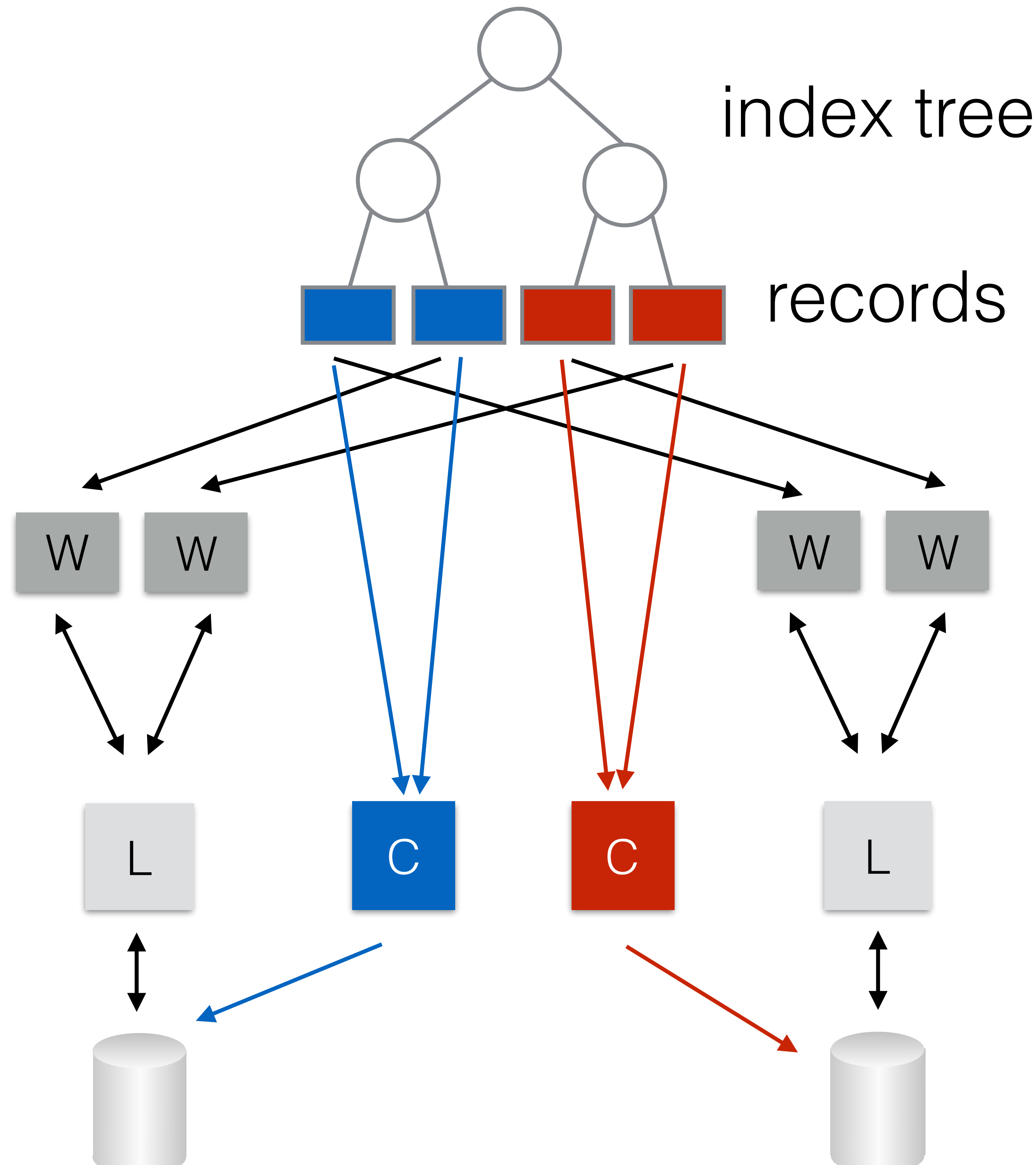


$\mathbf{e_P} = \min \{e_1, e_2, e_3, e_4\} - 1$

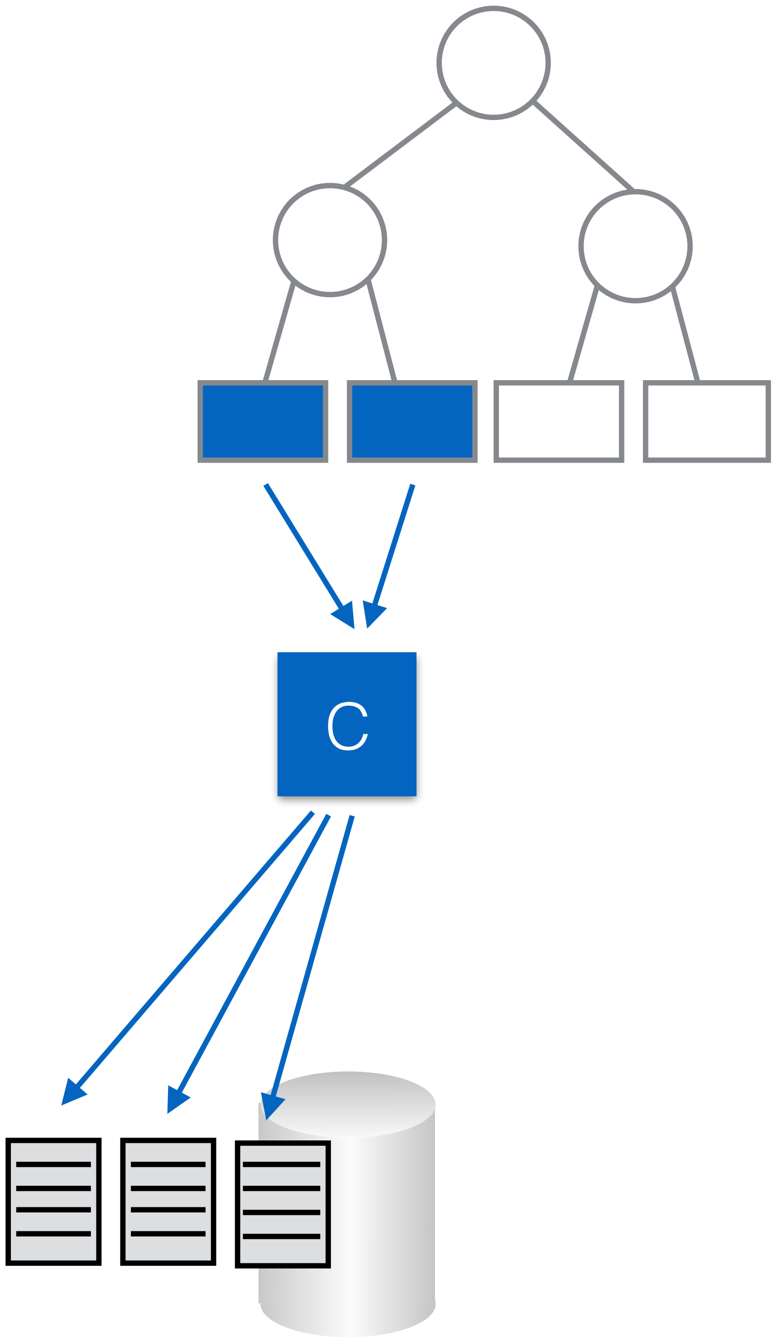
all transactions in epochs <= $\mathbf{e_P}$ are persistent

- Silo Overview
- **SiloR Design**
    - Logging
    - **Checkpointing**
    - Recovery
- Evaluation
- Related work

- Parallel checkpointing
- Checkpoint happens regularly



- Tree walk over a range of each table - inconsistent checkpoint
- Only committed records in checkpoint
- Writes out to multiple files, enabling <u>easy recovery parallelism</u>



Checkpoint

- Checkpoint starts in epoch $\mathbf{e_L}$
  - skips over records with TID.**e** such that $\mathbf{e \geq e_L}$

Checkpoint

- Checkpoint starts in epoch $\mathbf{e_L}$
    - skips over records with TID.**e** such that **e >=** $\mathbf{e_L}$
    - <u>smaller checkpoints -> smaller log -> faster recovery</u>

Checkpoint

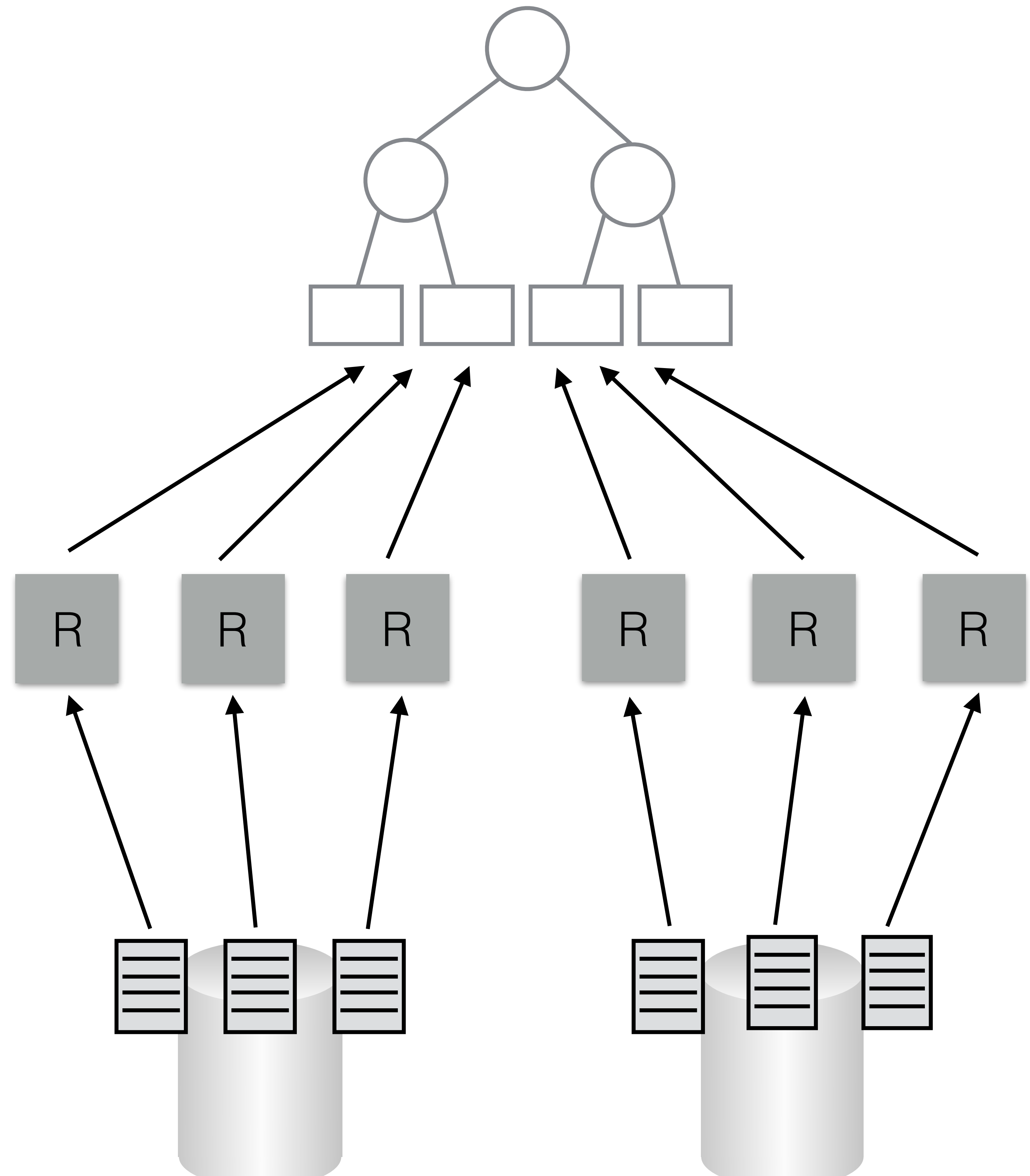
- Checkpoint starts in epoch $\mathbf{e_L}$
  - skips over records with TID.**e** such that $\mathbf{e >= e_L}$
  - <u>smaller checkpoints -> smaller log -> faster recovery</u>
- Checkpoint ends in epoch $\mathbf{e_H}$
  - usable once $\mathbf{e_H} <= \mathbf{e_P}$
  - removes **old_data.e** log file where $\mathbf{e} < \mathbf{e_L}$

- Silo Overview
- **SiloR Design**
  - Logging
  - Checkpointing
  - **Recovery**
- Evaluation
- Related work

Recovery parallelism is easy because of our logging and checkpointing designs

Checkpoint recovery

Easy parallelism:
one checkpoint
recovery
thread per file



Log Recovery

- Value logging enables log files to be played in *any order* — highest TID per key wins

Log Recovery

- Value logging enables log files to be played in *<u>any order</u>* — highest TID per key wins
  - logs in later epochs replayed first

Log Recovery

- Value logging enables log files to be played in *any order* — highest TID per key wins
  - logs in later epochs replayed first
- No log record from **epoch > $\mathbf{e_p}$** is replayed

- Silo Overview
- SiloR Design
  - Logging
  - Checkpointing
  - Recovery
- **Evaluation**
- Related work

Evaluation

- Experiment setup
  - single machine with four 8 core Intel Xeon E7-4830 processors (32 physical cores)
  - machine has 256 GB of DRAM, 64 GB of DRAM attached to each socket
  - 4 disks: 3 Fusion IO drives, 1 RAID-5 disk array

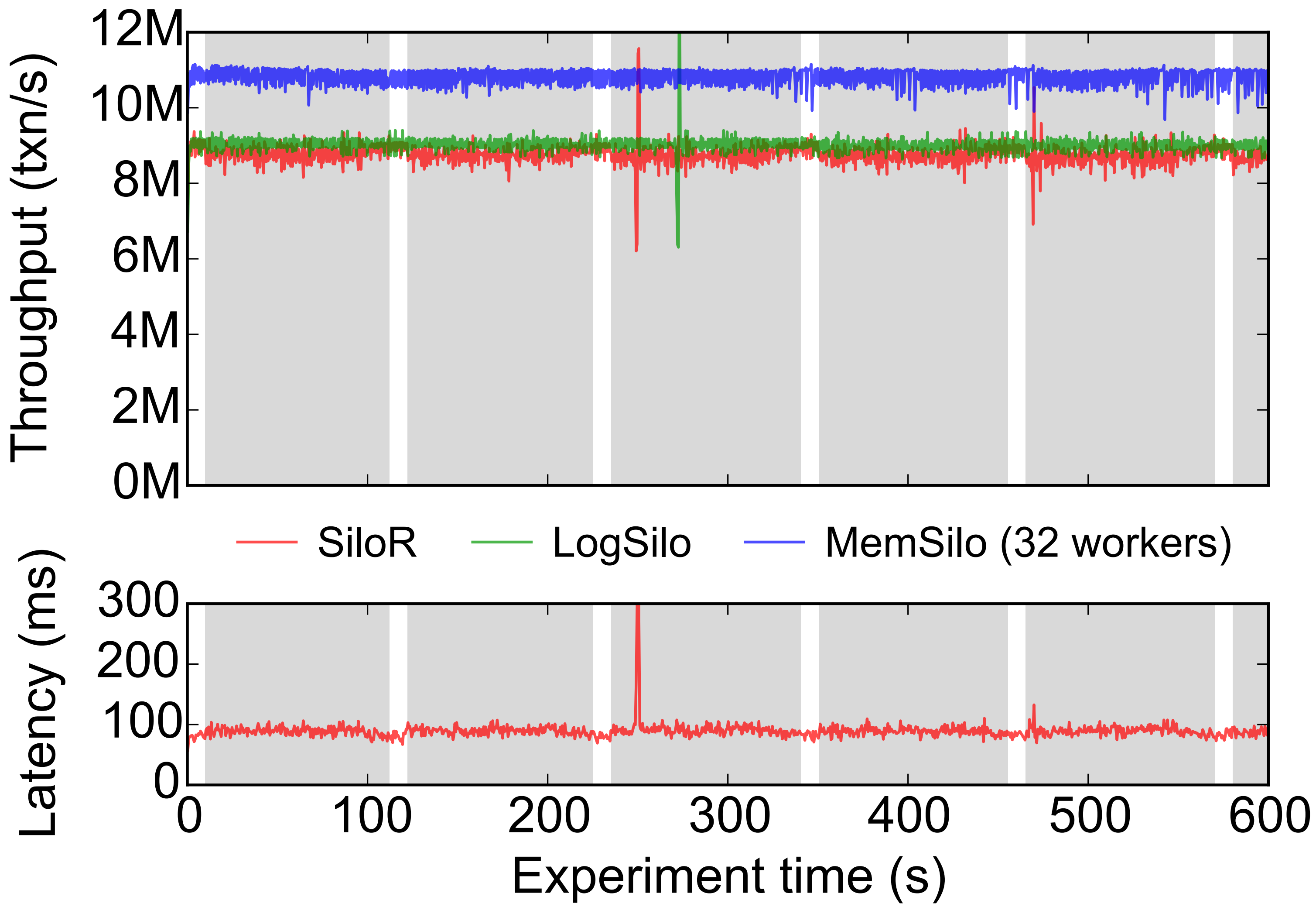
Evaluation Goals

- Can SiloR keep up with high transaction throughput from Silo?
- Does recovery take no more than a few minutes for a large database?

Evaluation - YCSB-A

- Key-value benchmark
- 400 million keys, 100 byte records
- 70% read, 30% write
- 28 workers, 4 loggers, 4 checkpoint threads
- Database does not grow

Evaluation - YCSB-A



Avg throughput: 8.76 Mtxns/s, 9.01 Mtxns/s, 10.83 Mtxns/s

Recovery for YCSB-A

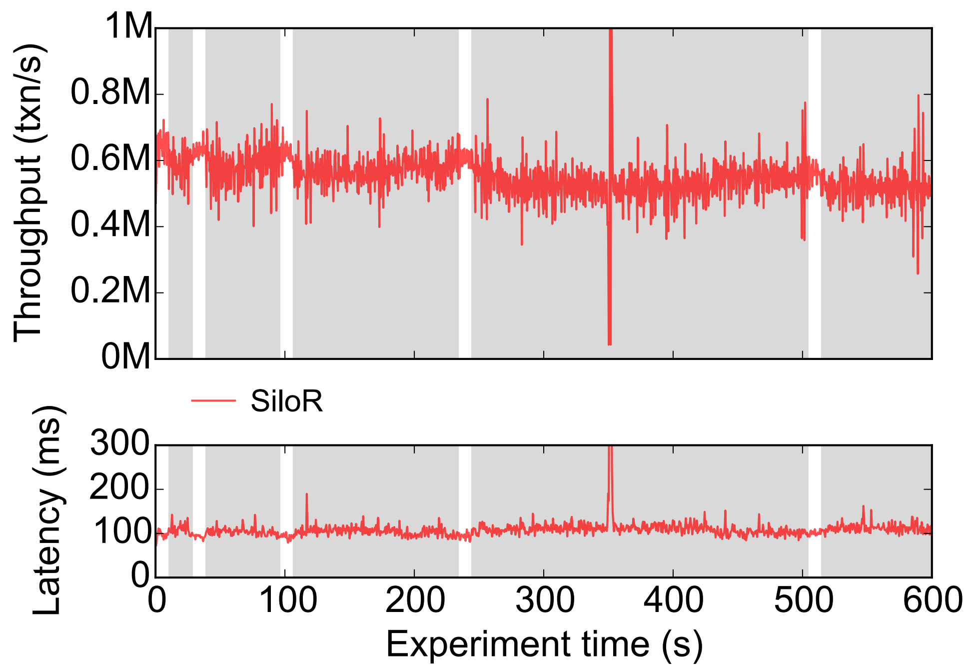
Simulates crash right before the *second* checkpoint completes

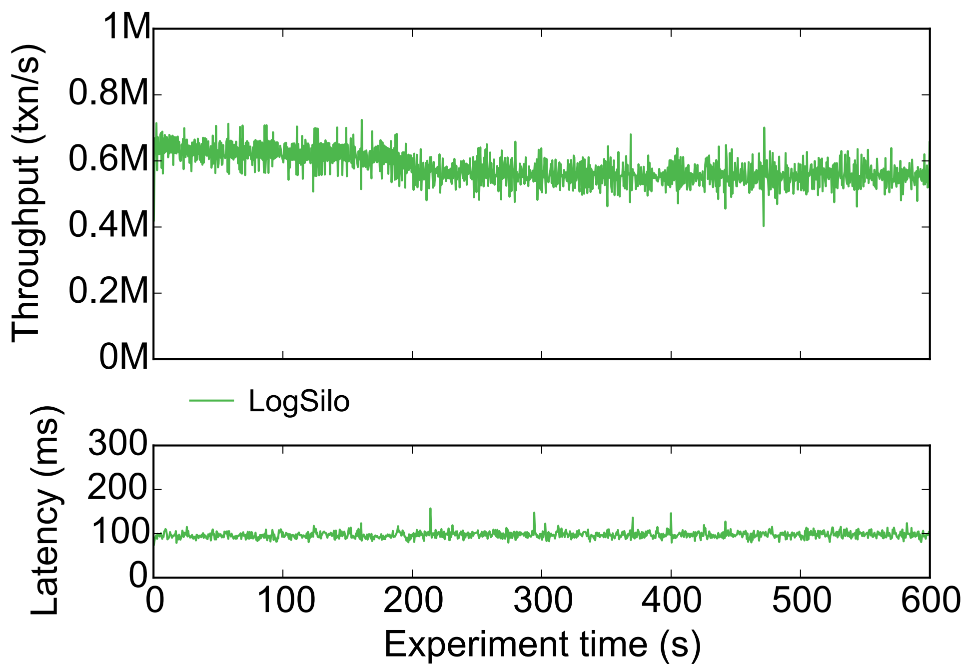
	Recovered database	Checkpoint	Log	Total
Size	43.2 GB	36 GB	64 GB	100 GB
Recovery time		33 s	73 s	106 s

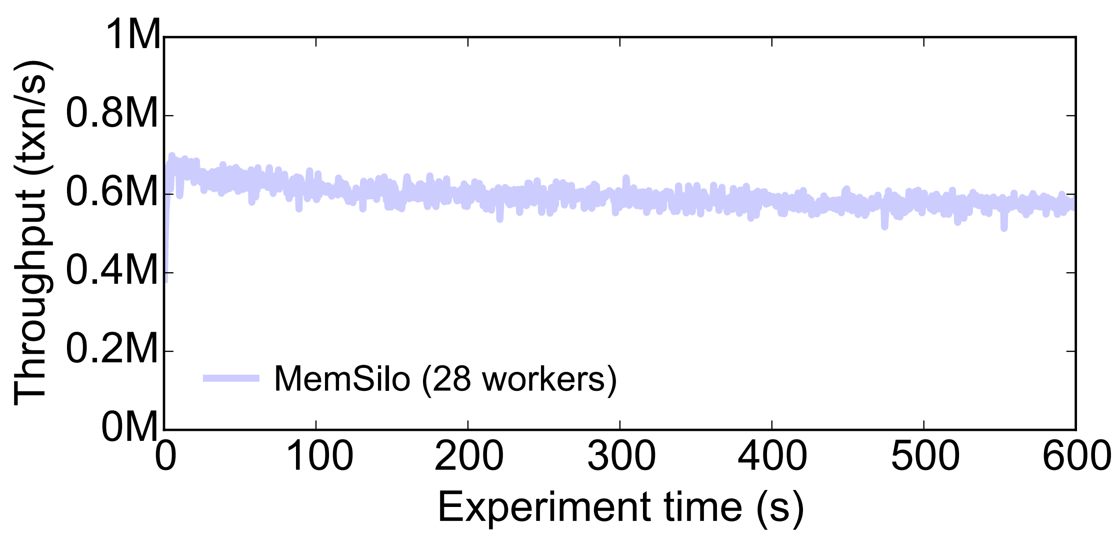
Evaluation - TPC-C

- TPC-C is a popular OLTP benchmark
- 28 workers, 4 loggers, 4 checkpoint threads
- Database size grows very fast
- Checkpoint period also grows

Evaluation - TPC-C







Avg throughput: 548 Ktxns/s, 575 Ktxns/s, 592 Ktxns/s

Recovery for TPC-C

Simulates crash right before the *fourth* checkpoint completes

	Recovered tuples	Checkpoint	Log	Total
Size	72.2 GB	15.7 GB	180 GB	195.7 GB
Recovery time		17 s	194 s	211 s

Evaluation conclusion

Can SiloR keep up with high transaction throughput from Silo?

Does recovery take no more than a few minutes for a large database?

Evaluation conclusion

Can SiloR keep up with high transaction throughput from Silo?

Does recovery take no more than a few minutes for a large database?

- Silo Overview
- SiloR Design
  - Logging
  - Checkpointing
  - Recovery
- Evaluation
- **Related work**

Related work (partial list)

- VoltDB OLTP Recovery using command logging (ICDE ’14): operation logging advantages
- Recovery on RAMCloud (SOSP ’11): really fast recovery
- Fast checkpoint recovery on frequently consistent applications (SIGMOD ’11)

…

Conclusion

- Built a persistence system for a very fast multicore in-memory database
- Used parallelism in all parts of the system to enable
  - small degradation in runtime performance
  - recovery of large database in a few minutes

Questions?