MORE THAN YOU EVER WANTED TO KNOW ABOUT SYNCHRONIZATION

by Vincent Gramoli

ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming 2015

Mario Preishuber

Supervisor Christoph M. Kirsch

May 21, 2015

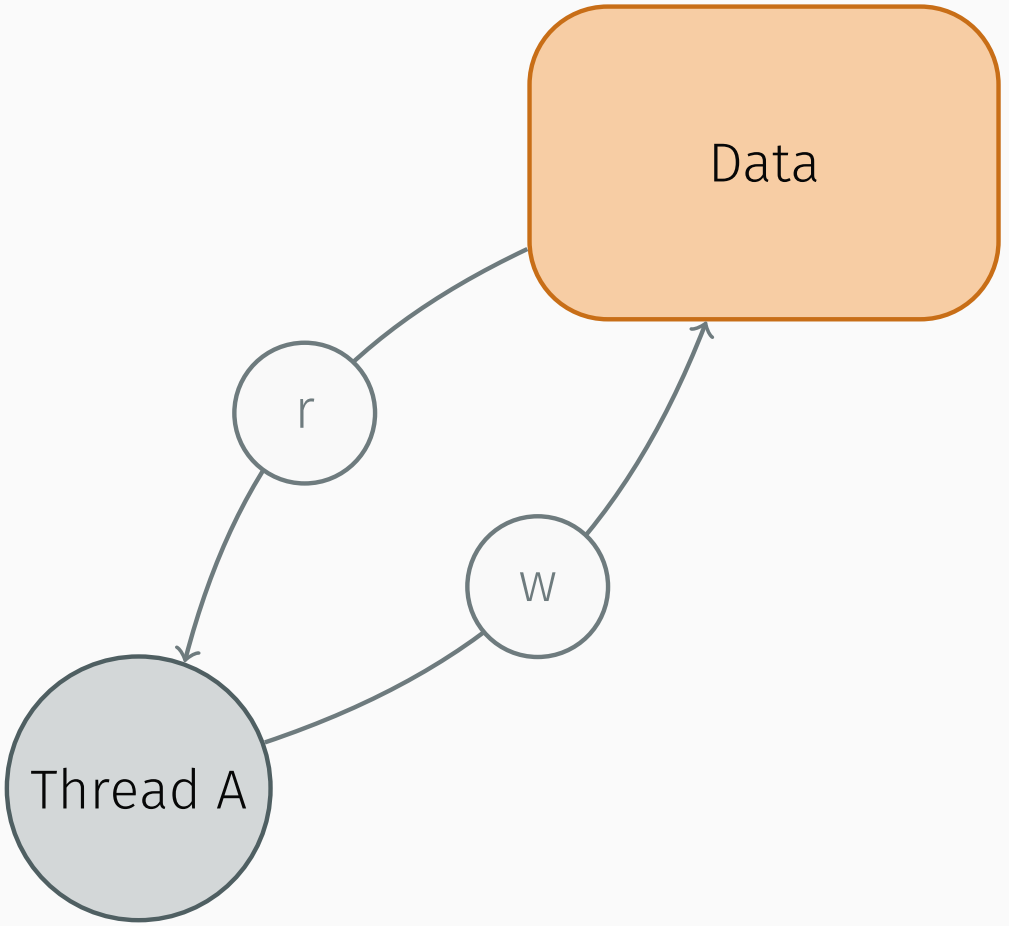
Concurrency & Memory Management Seminar 2015

Department of Computer Sciences

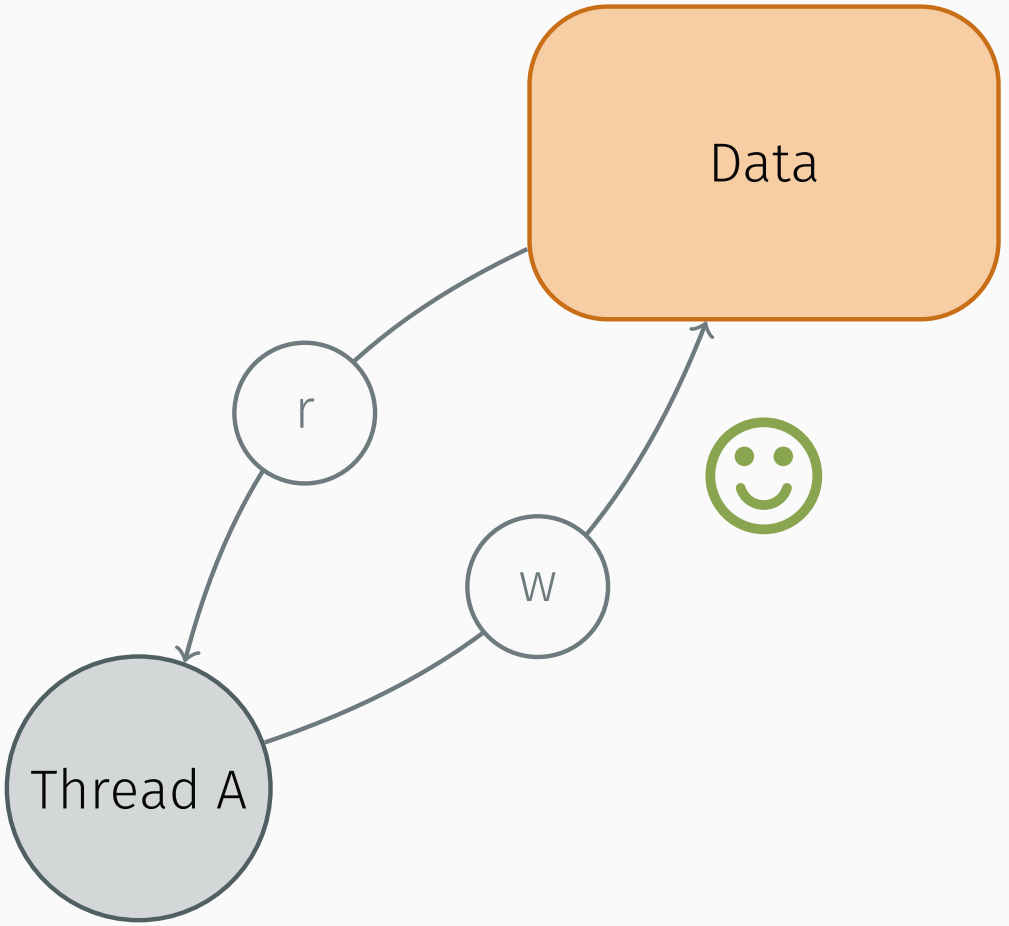
University of Salzburg

SYNCHRONIZATION

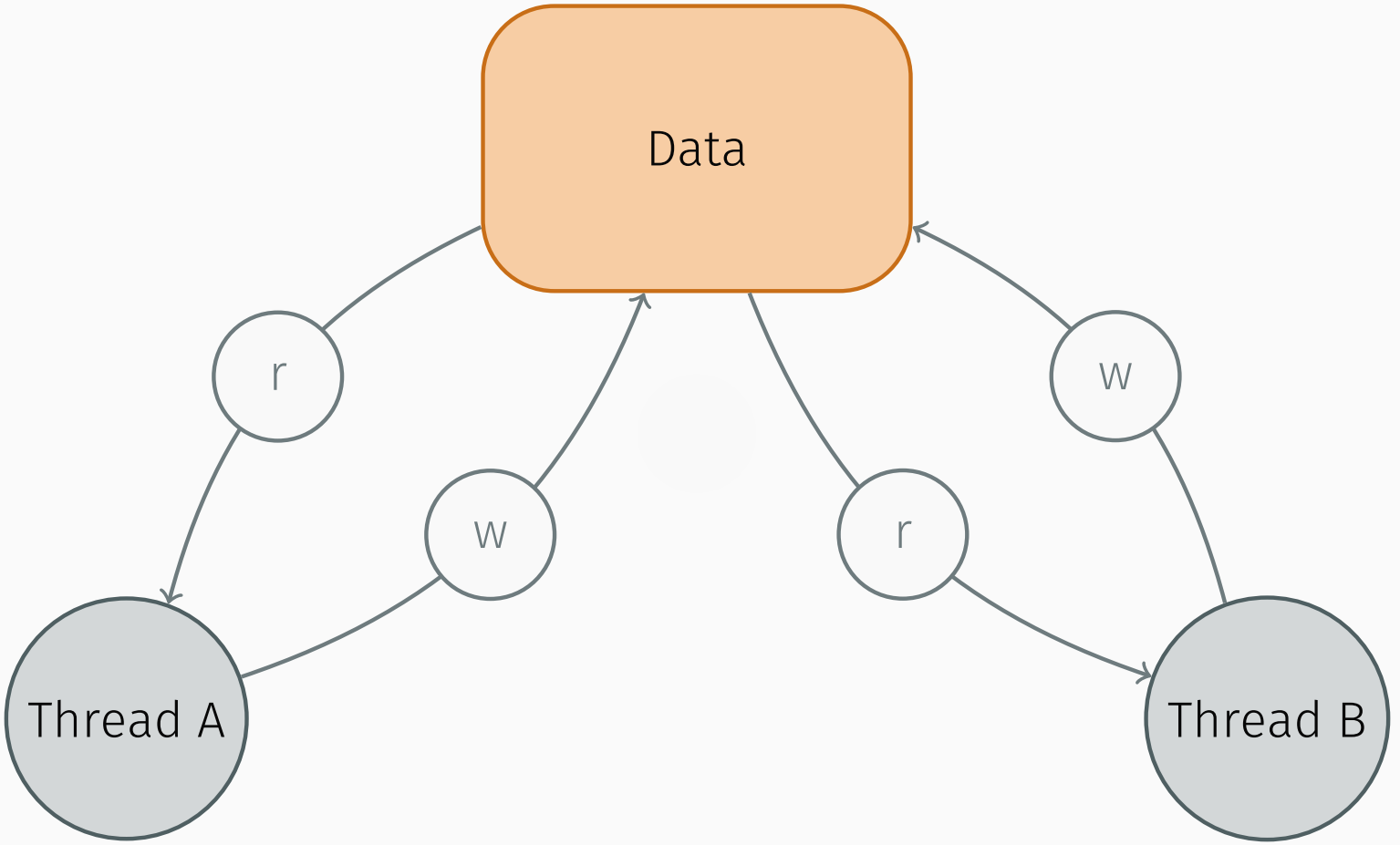
SYNCHRONIZATION



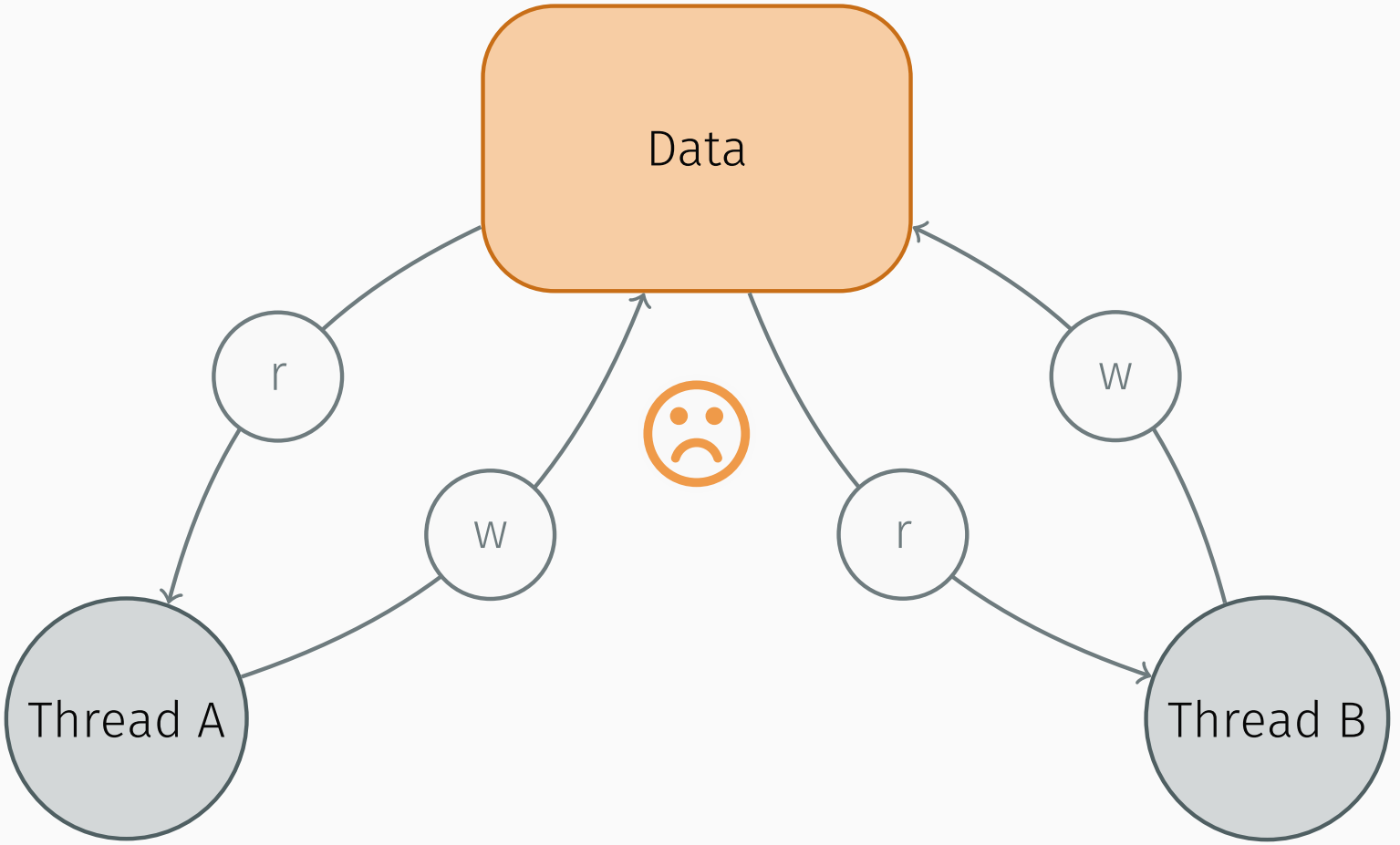
SYNCHRONIZATION



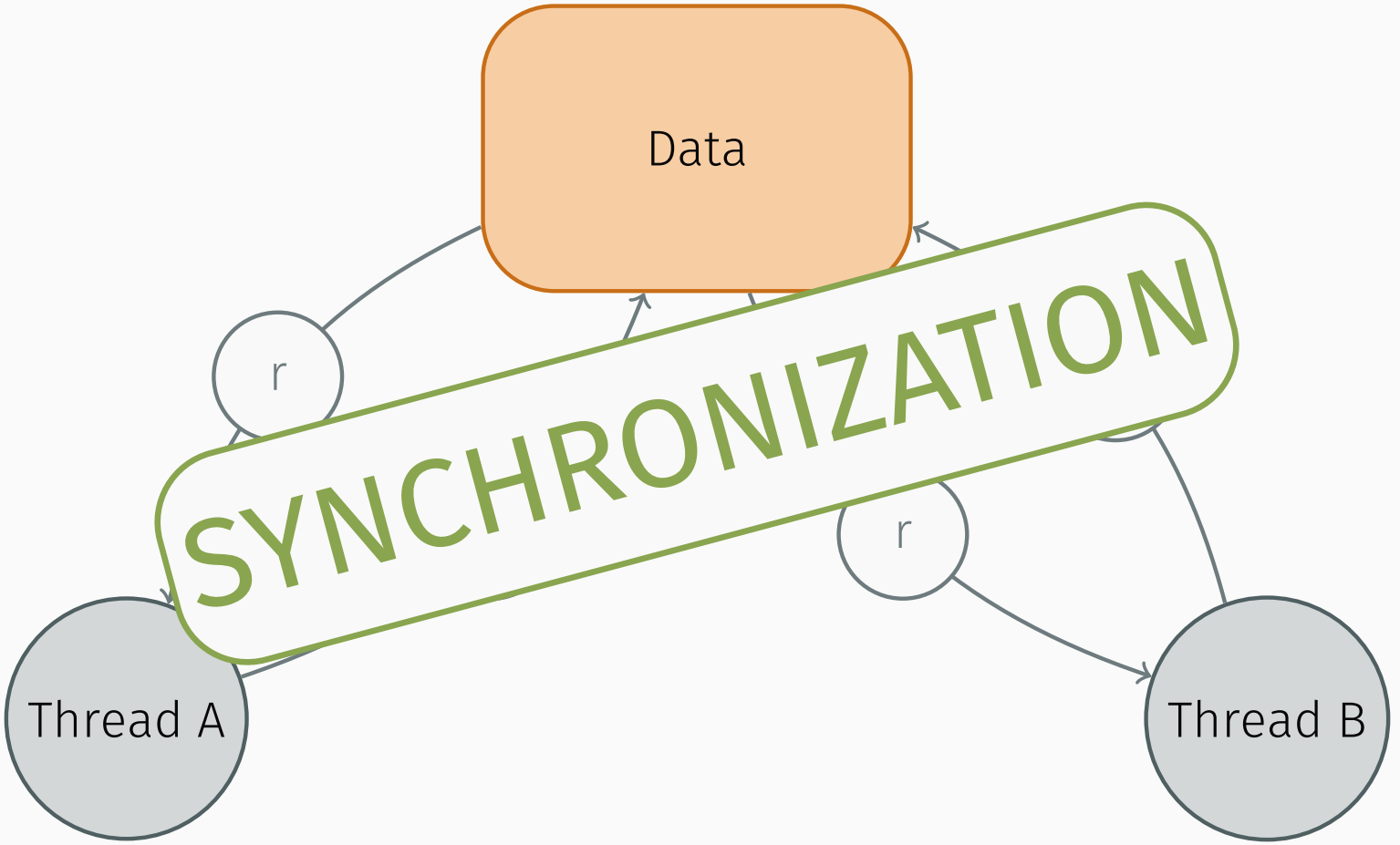
SYNCHRONIZATION



SYNCHRONIZATION

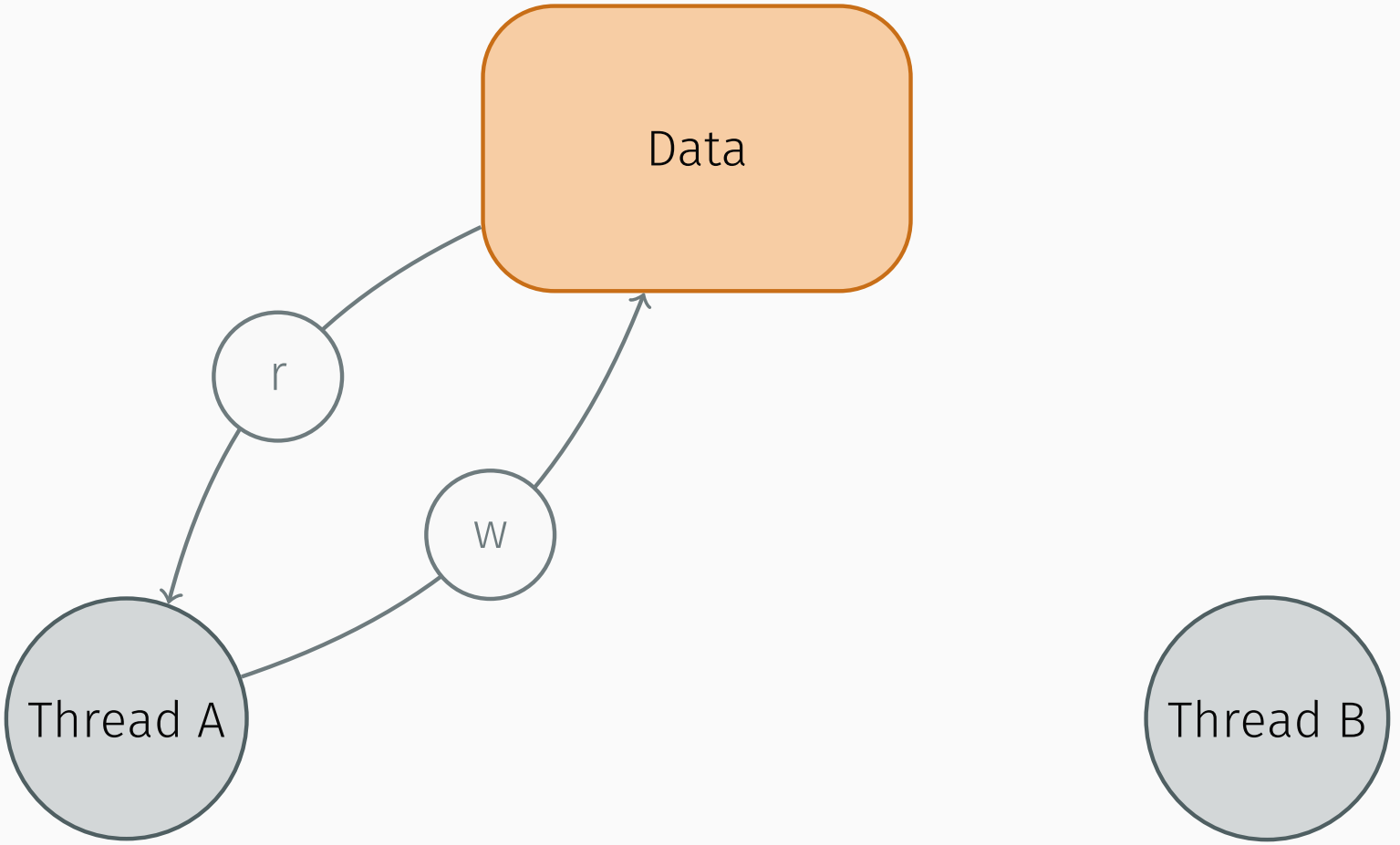


SYNCHRONIZATION

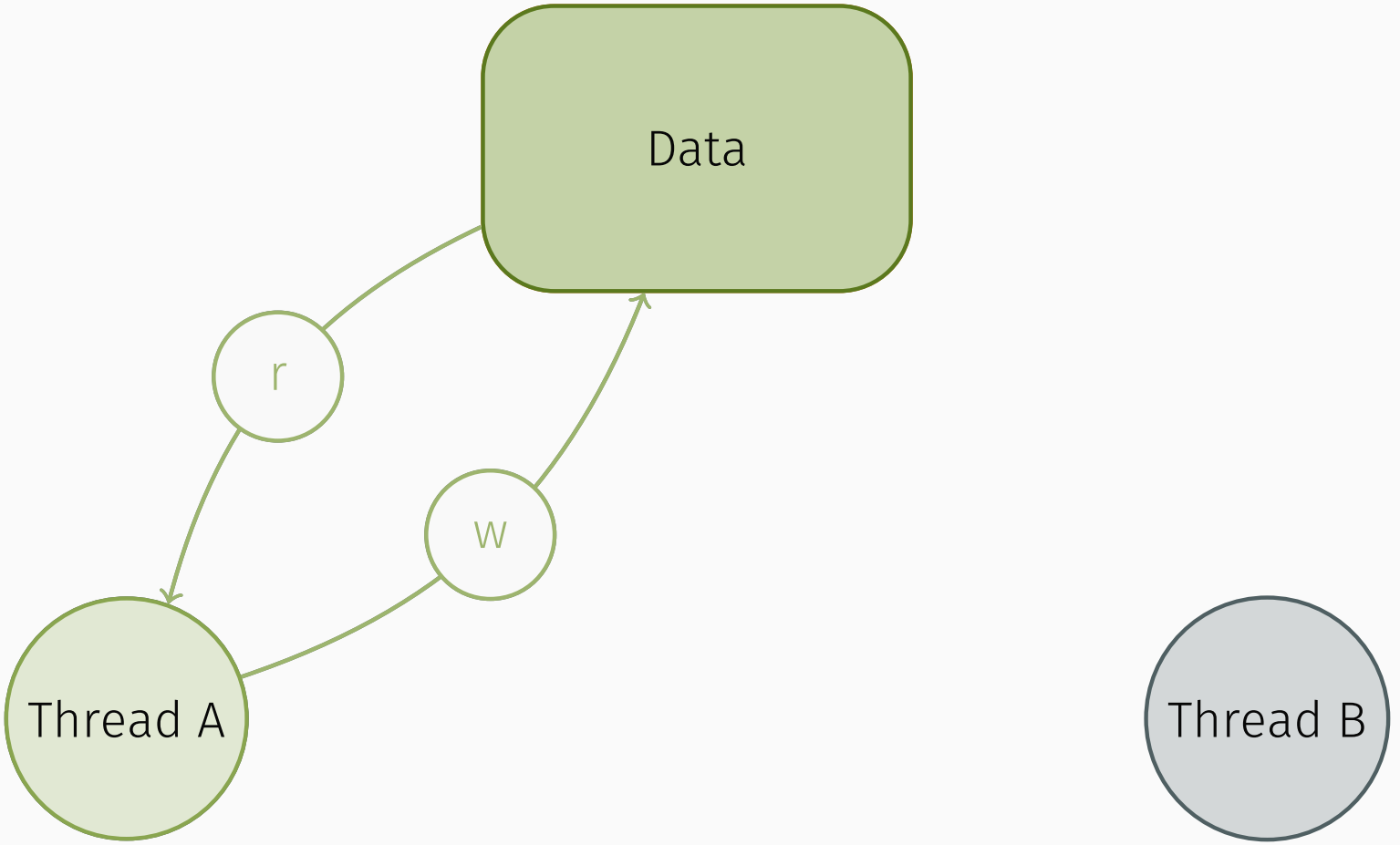


MUTUAL EXCLUSION

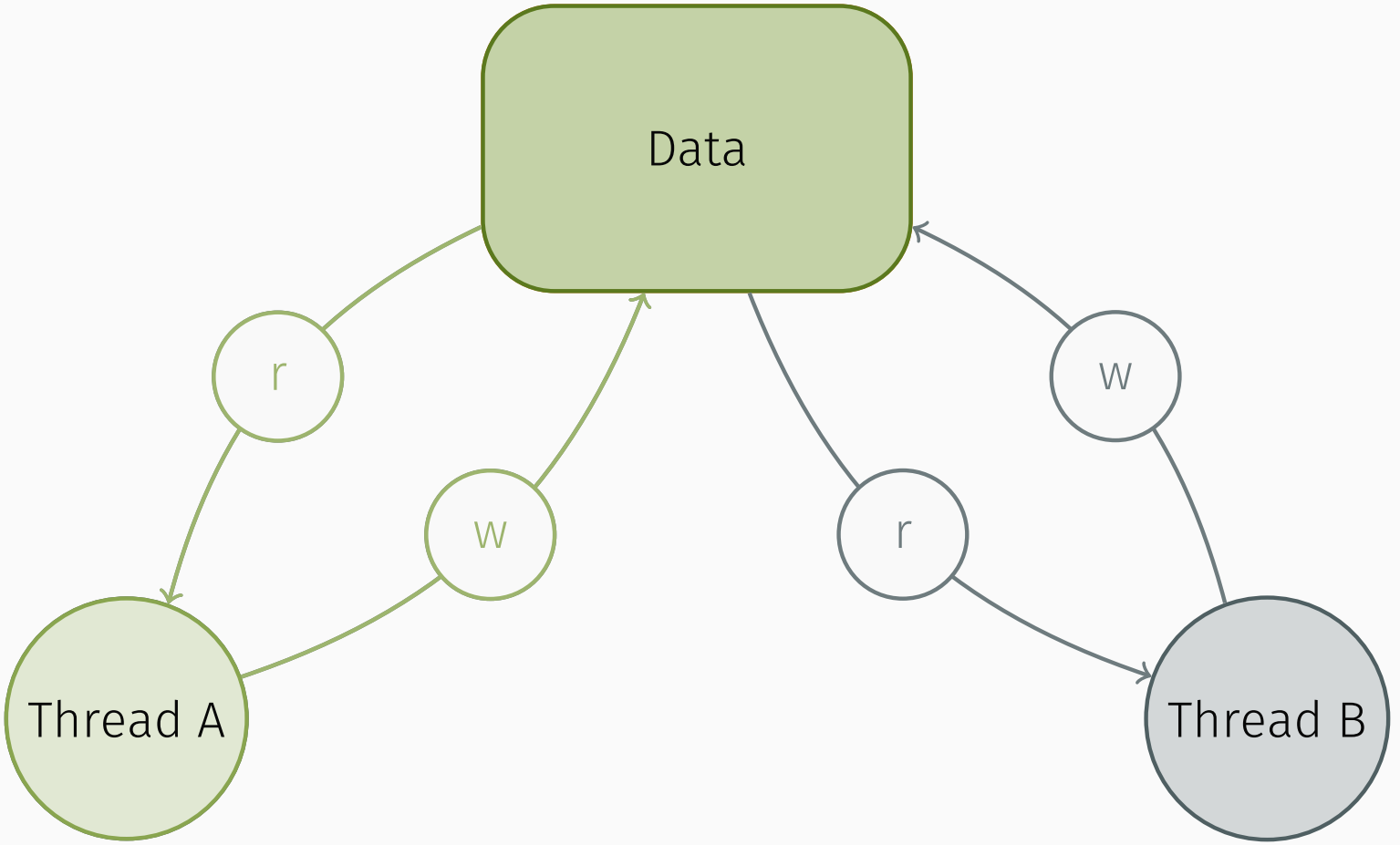
MUTUAL EXCLUSION



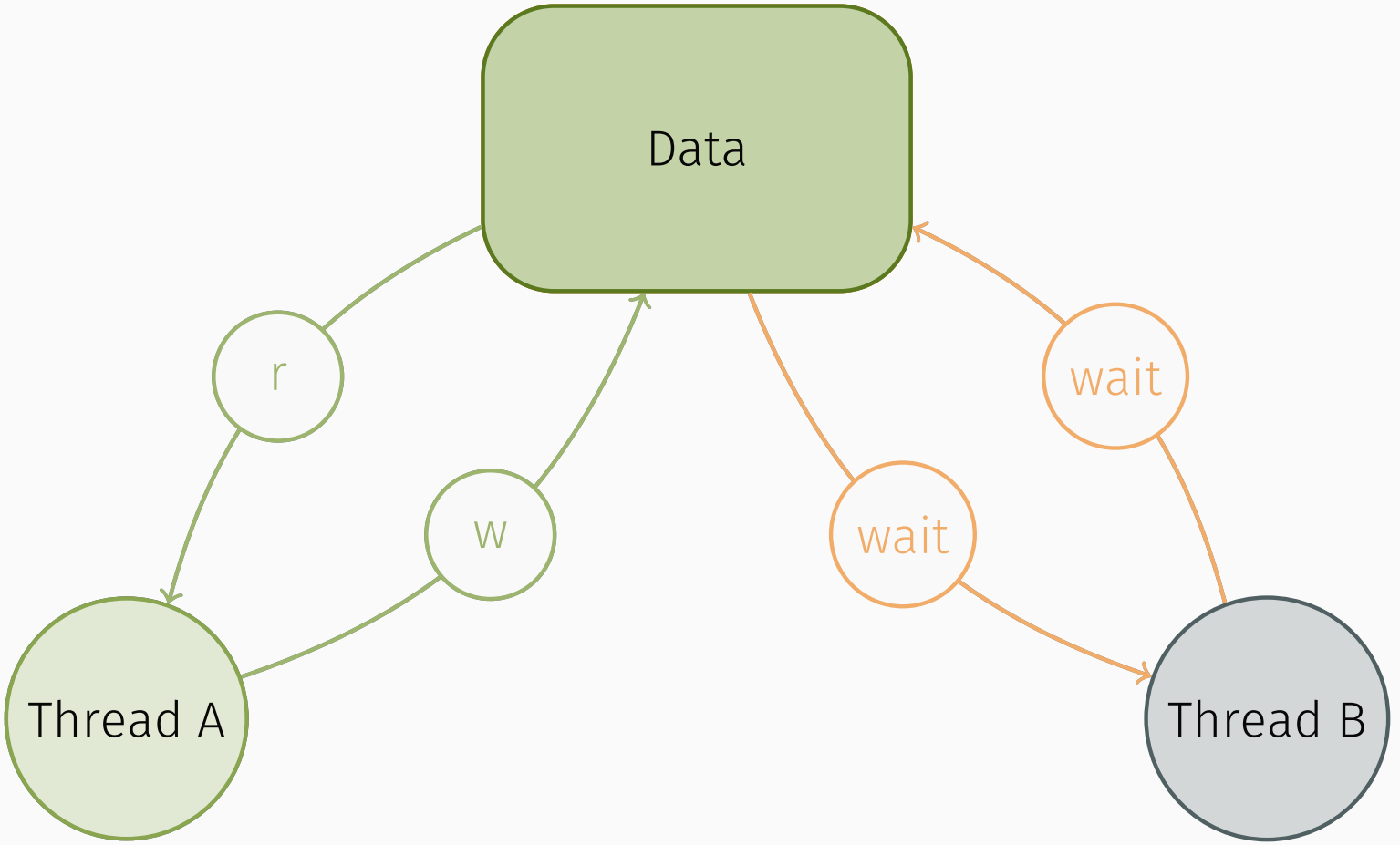
MUTUAL EXCLUSION



MUTUAL EXCLUSION

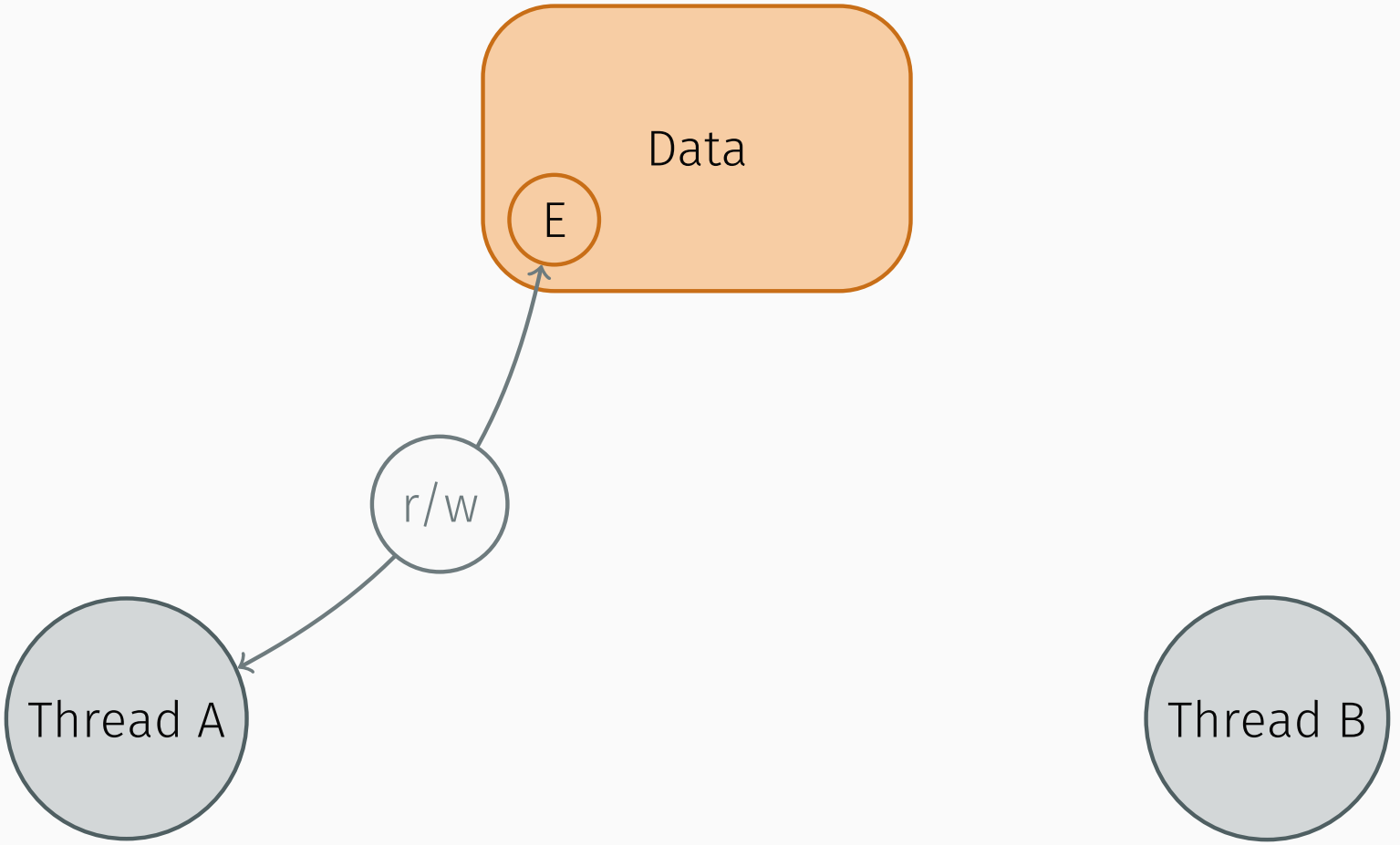


MUTUAL EXCLUSION

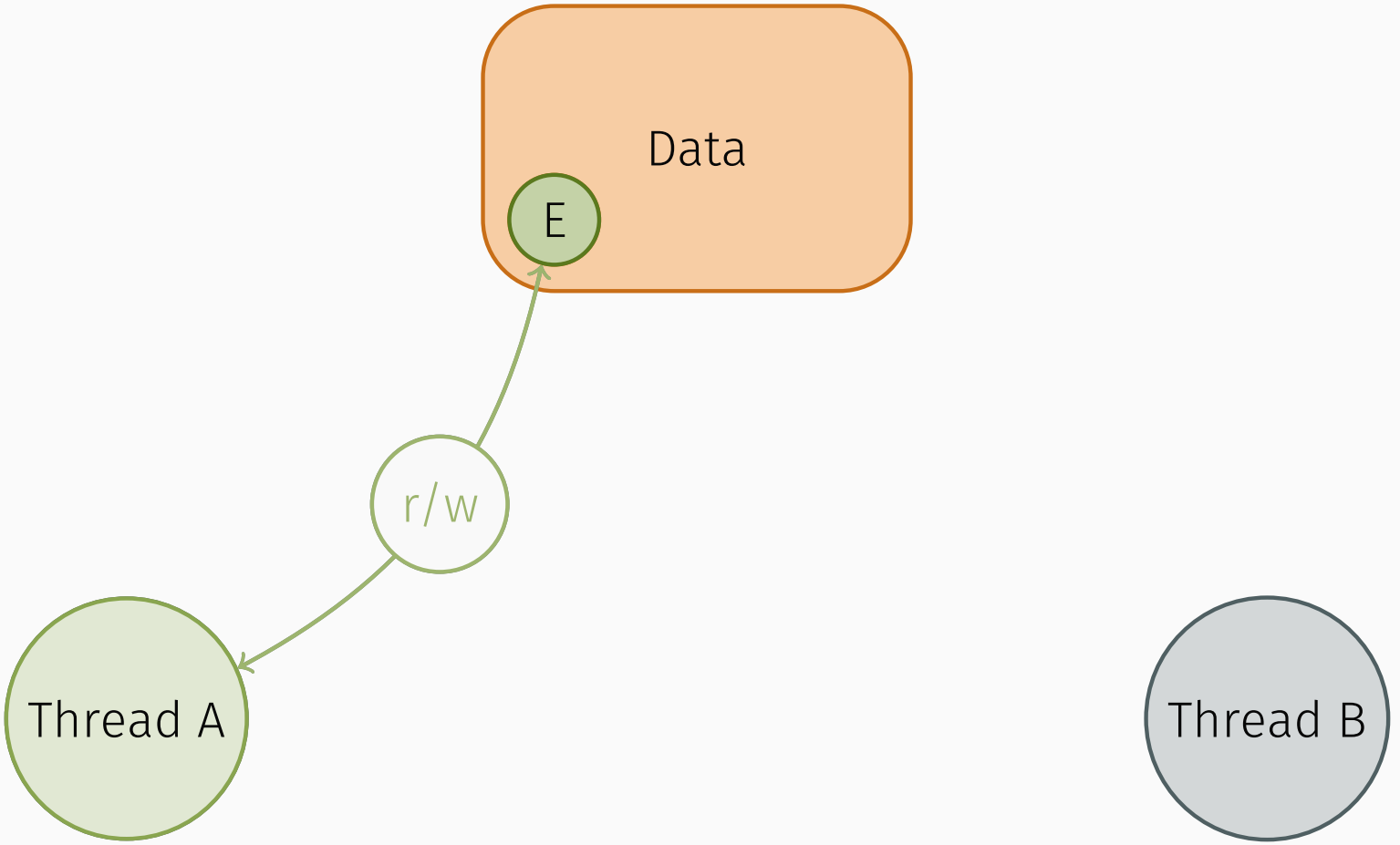


READ-MODIFY-WRITE

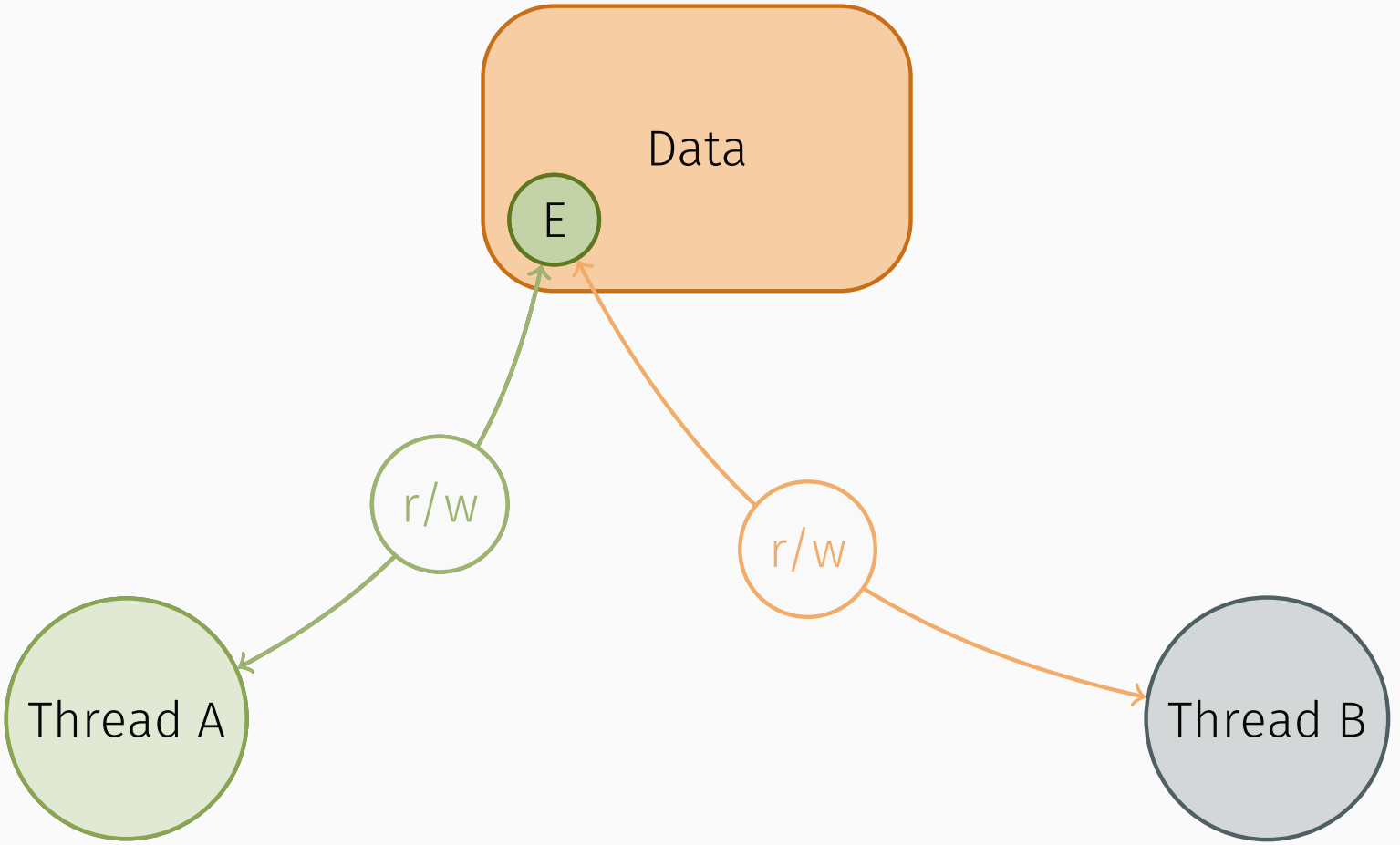
READ-MODIFY-WRITE



READ-MODIFY-WRITE

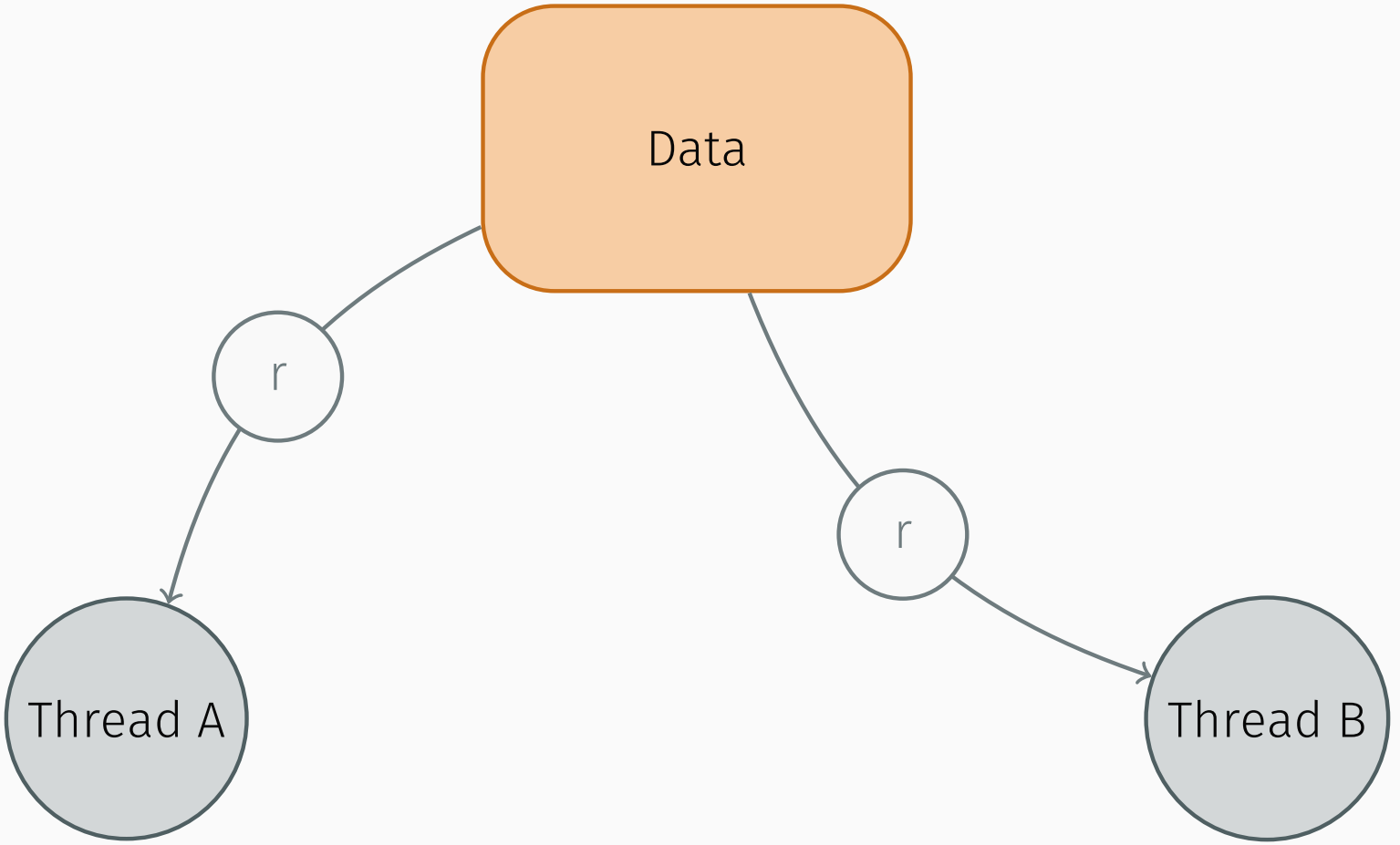


READ-MODIFY-WRITE

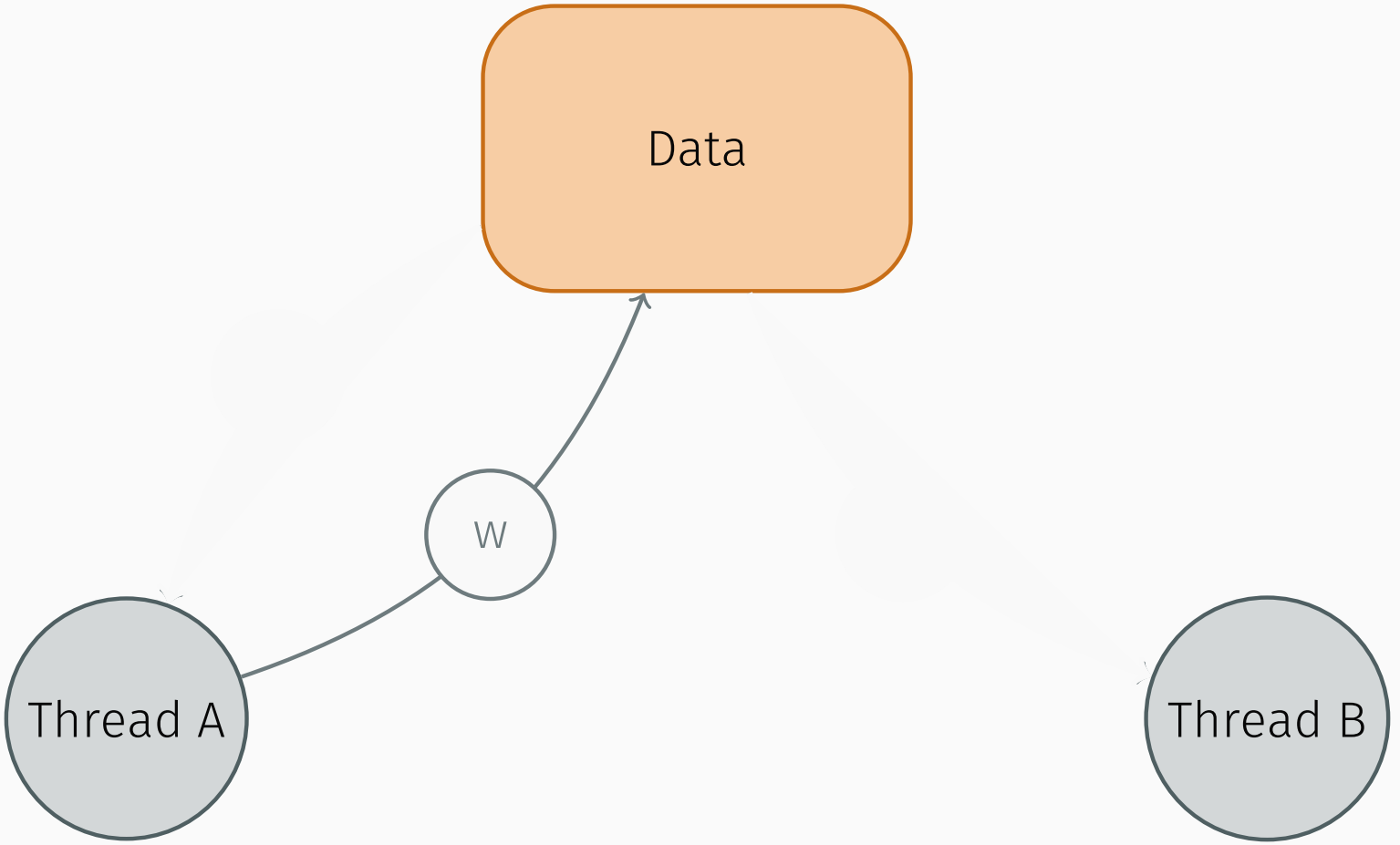


COPY-ON-WRITE

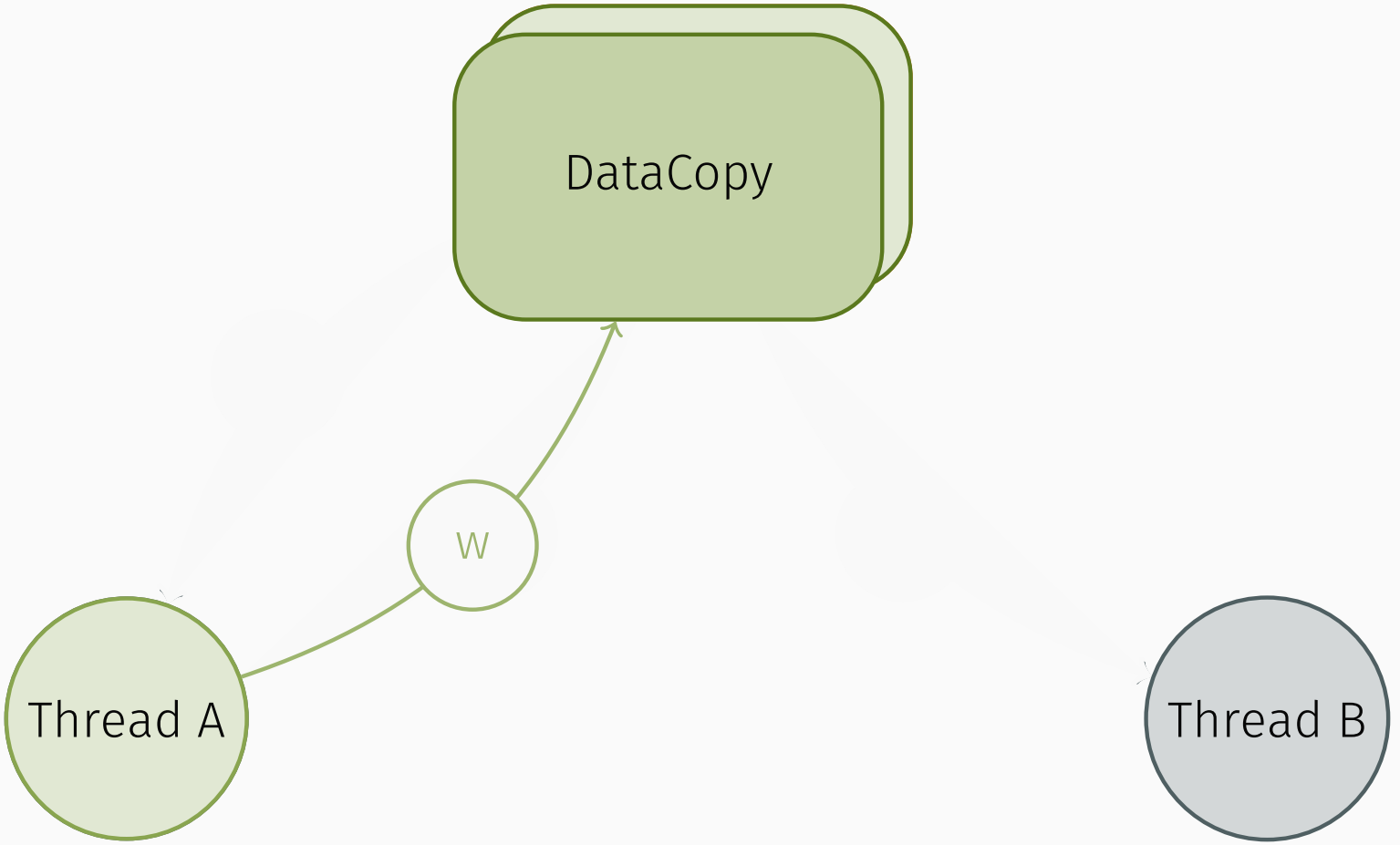
COPY-ON-WRITE



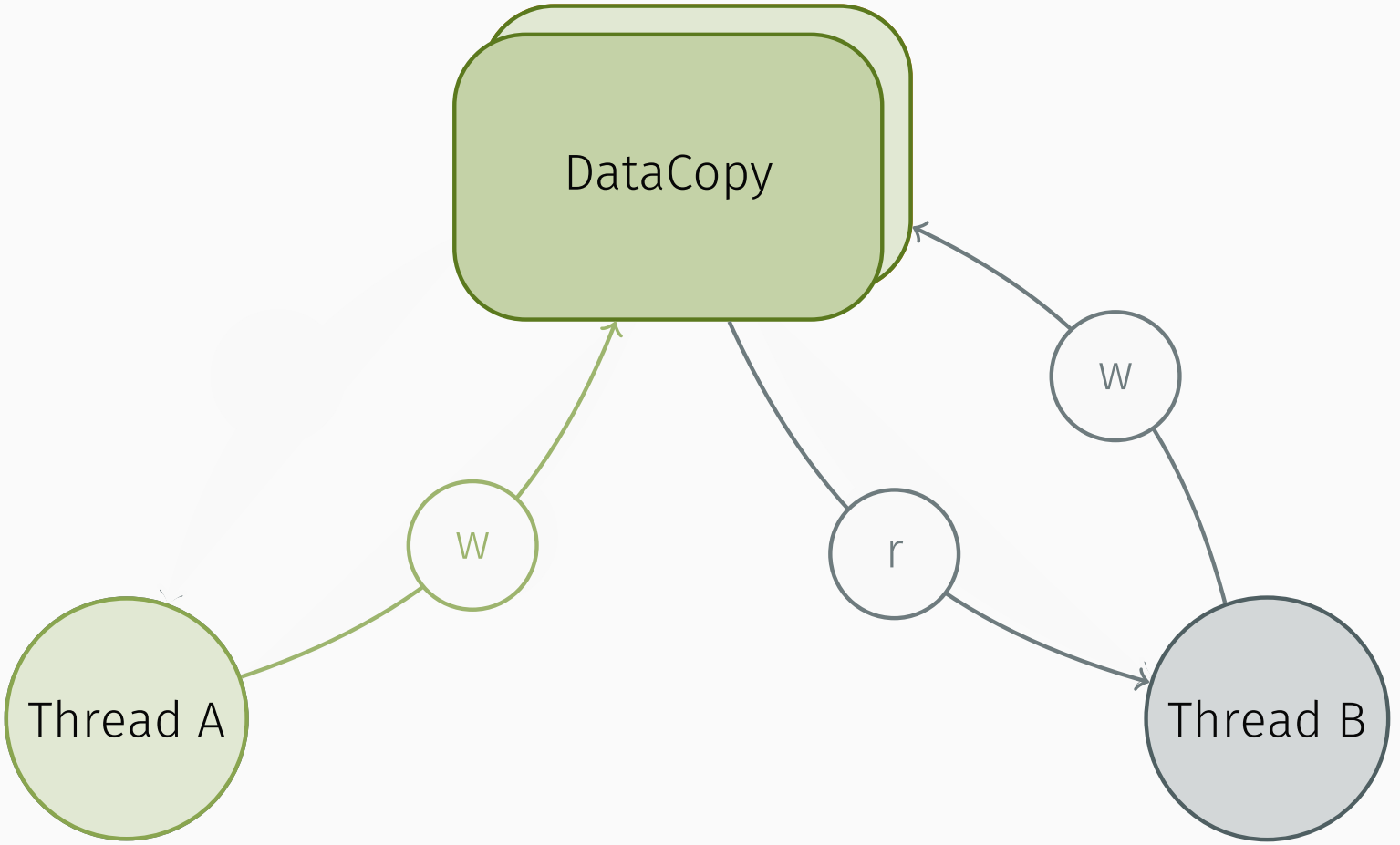
COPY-ON-WRITE



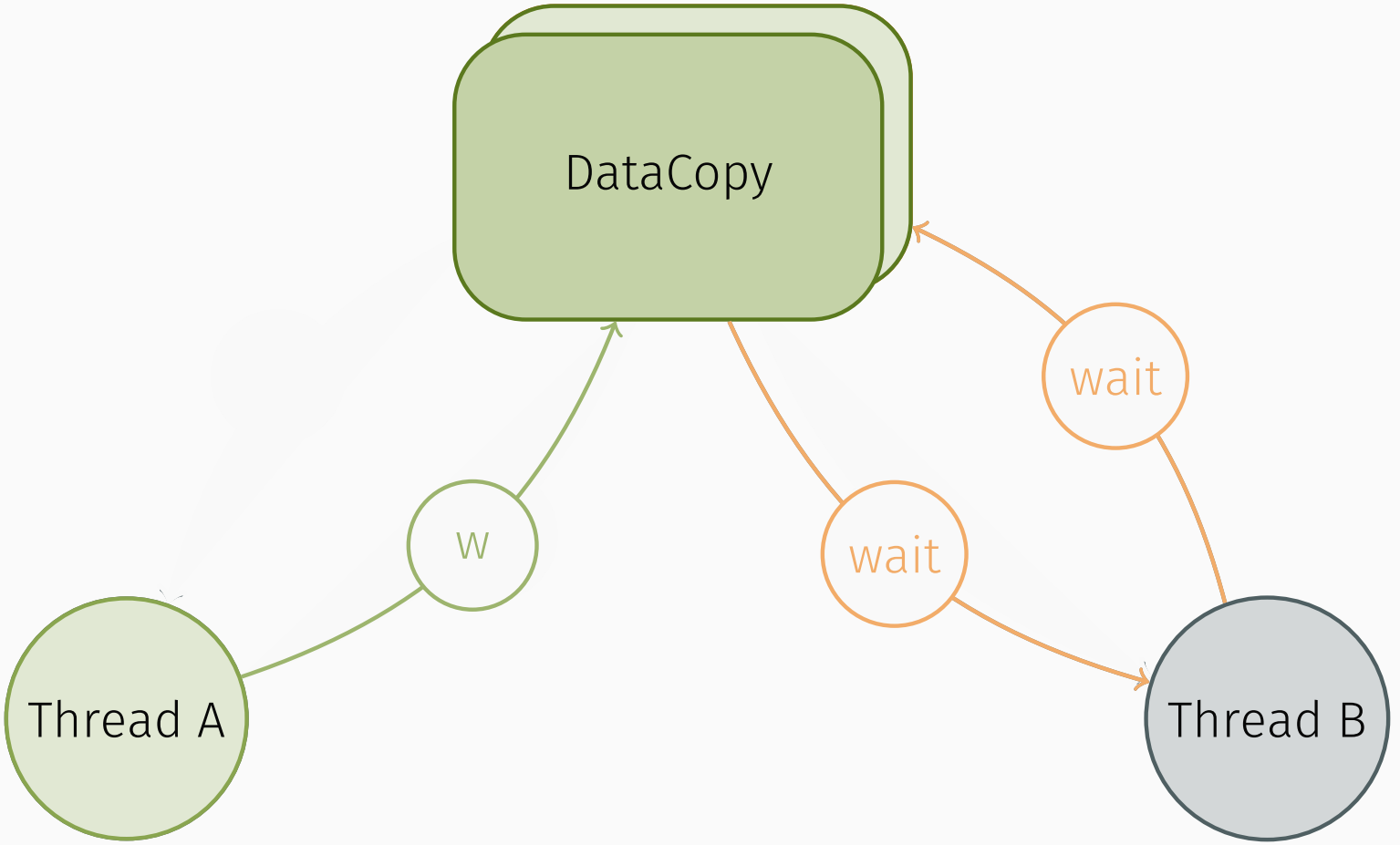
COPY-ON-WRITE



COPY-ON-WRITE

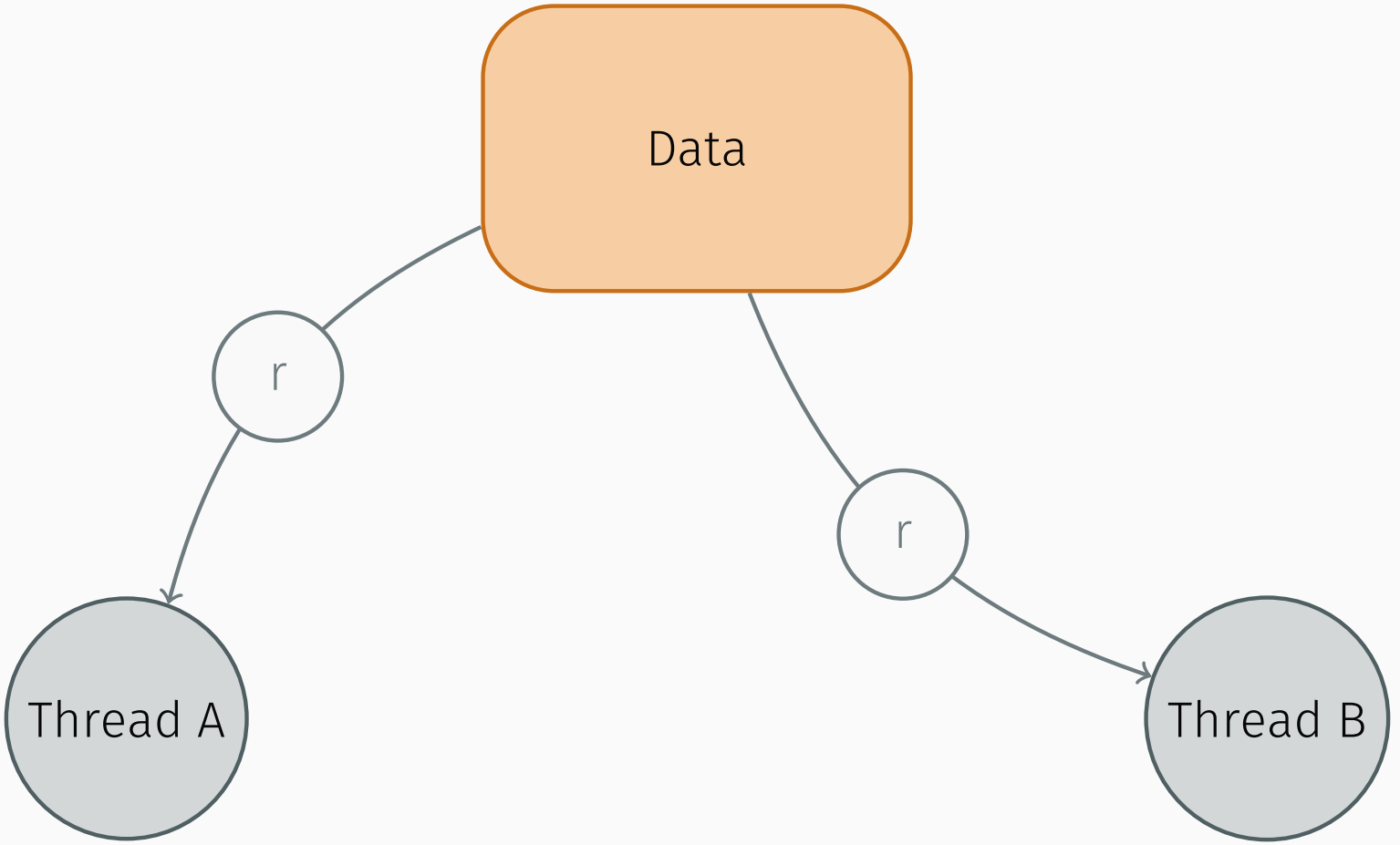


COPY-ON-WRITE

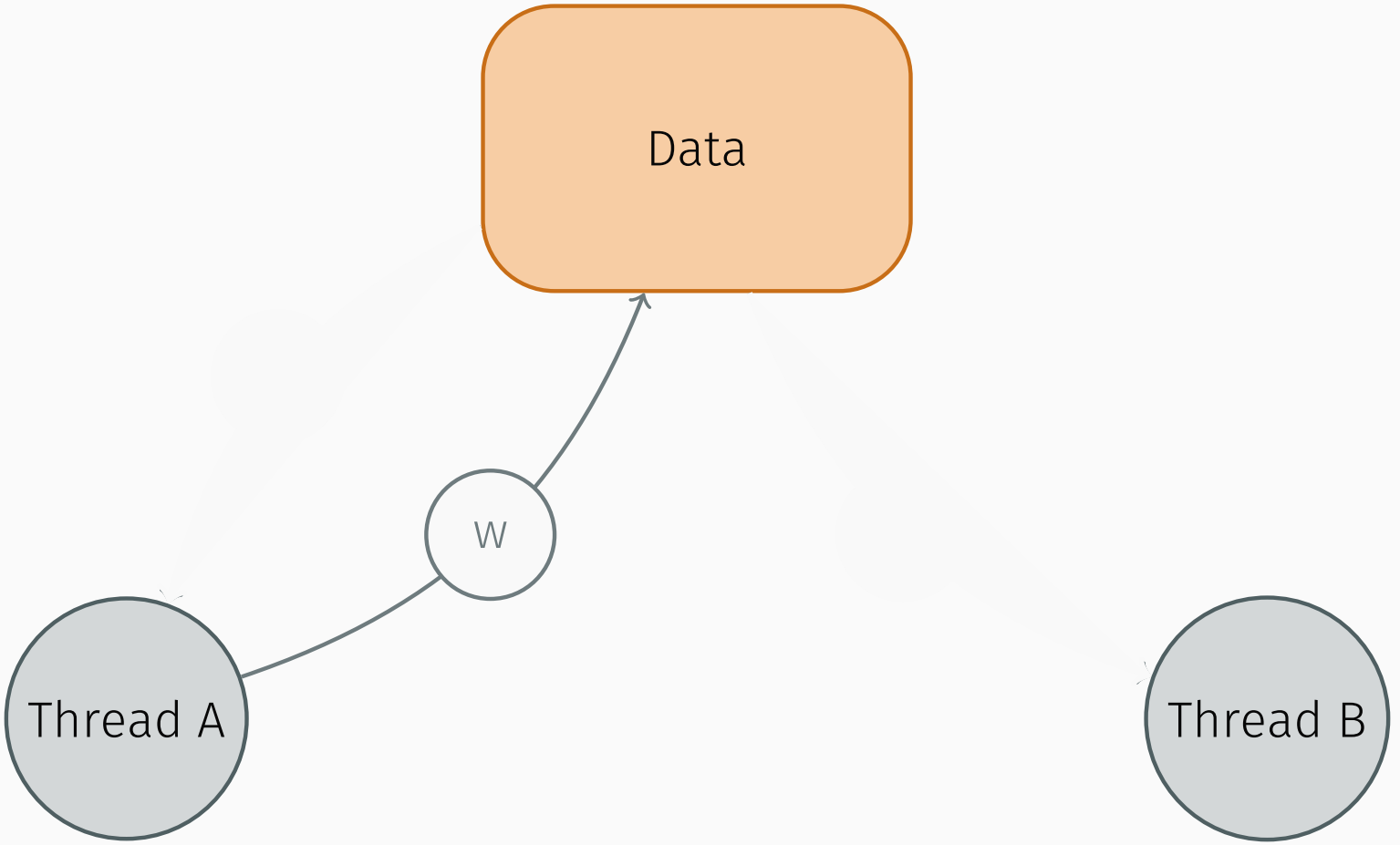


READ-COPY-UPDATE

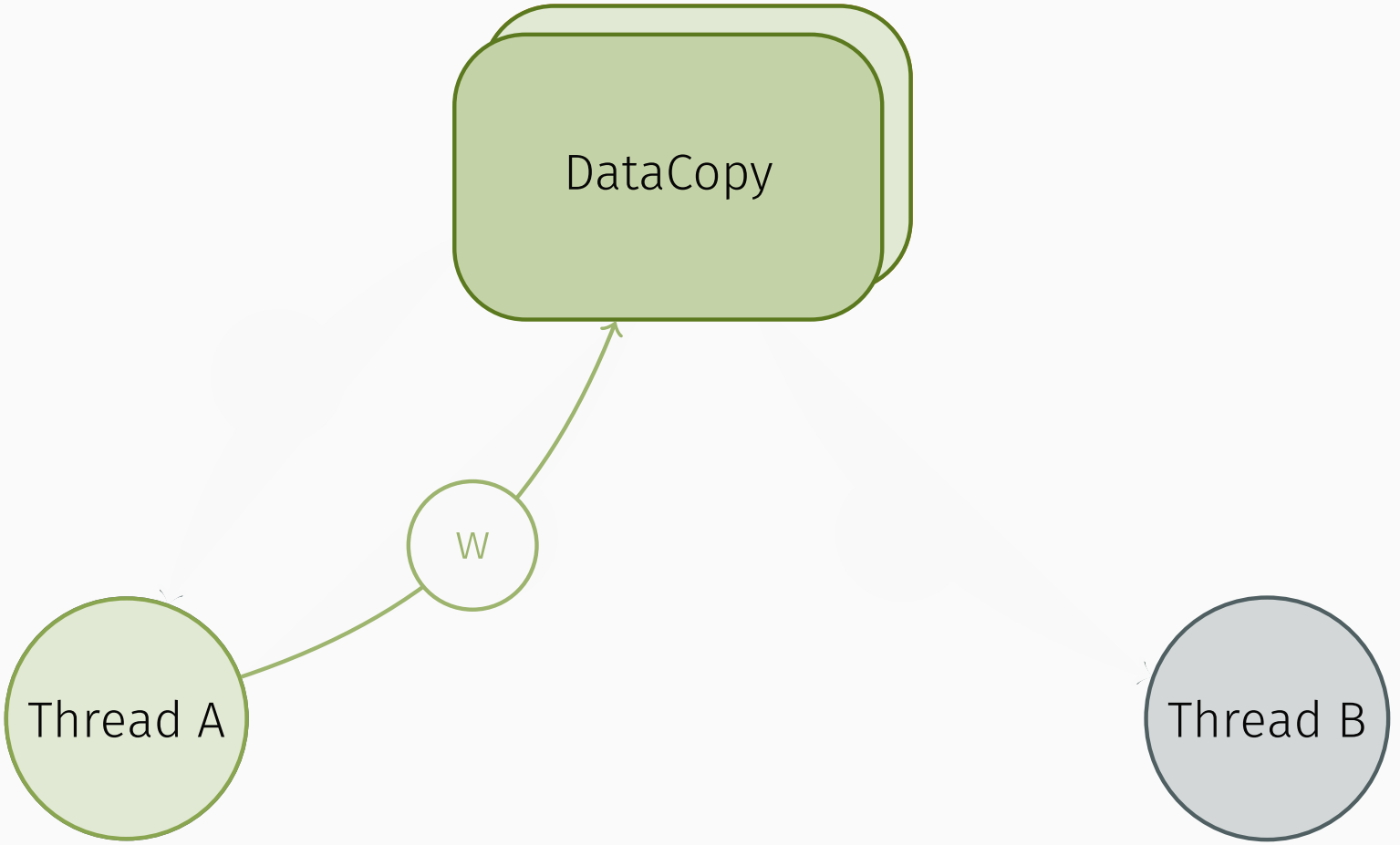
READ-COPY-UPDATE



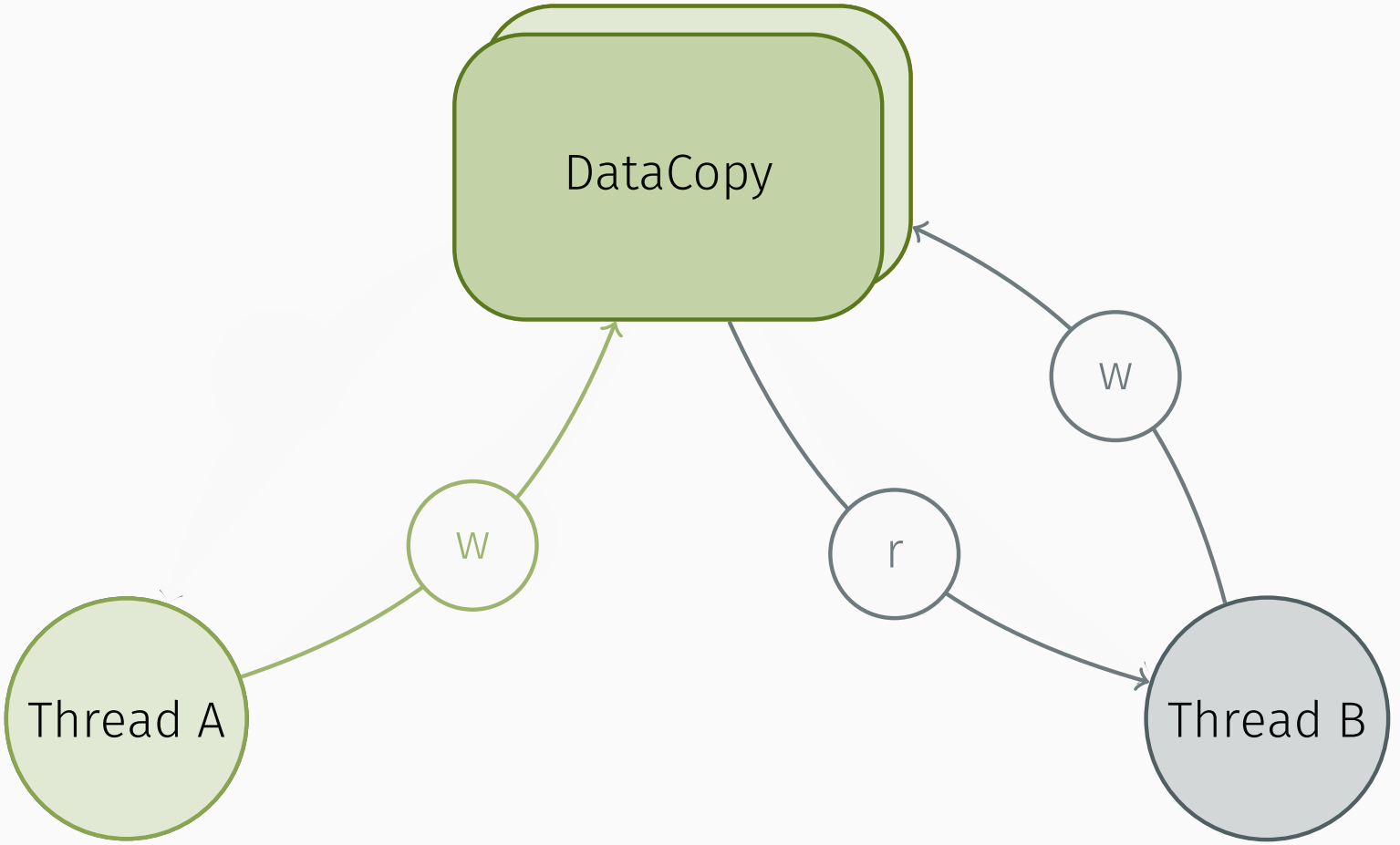
READ-COPY-UPDATE



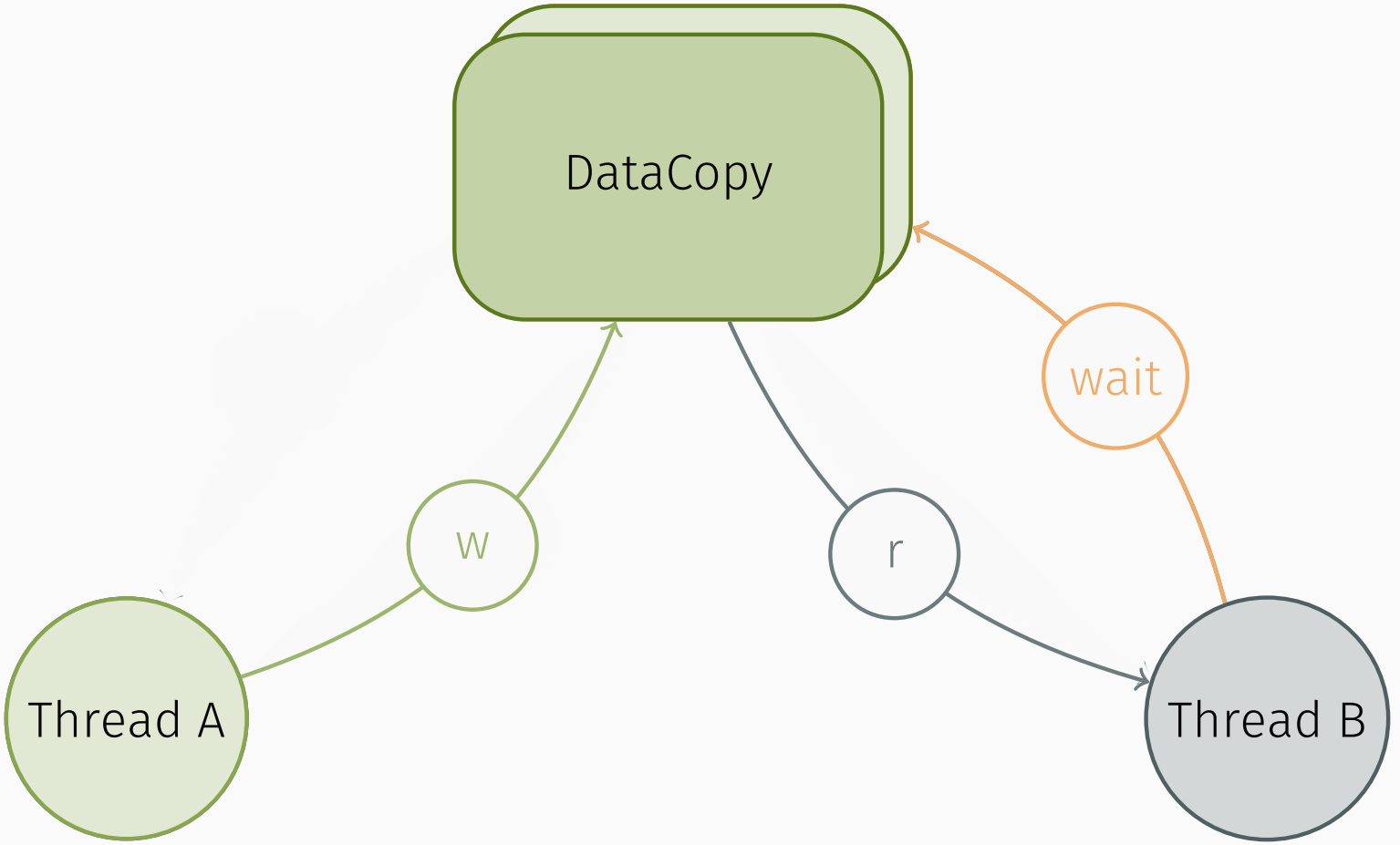
READ-COPY-UPDATE



READ-COPY-UPDATE

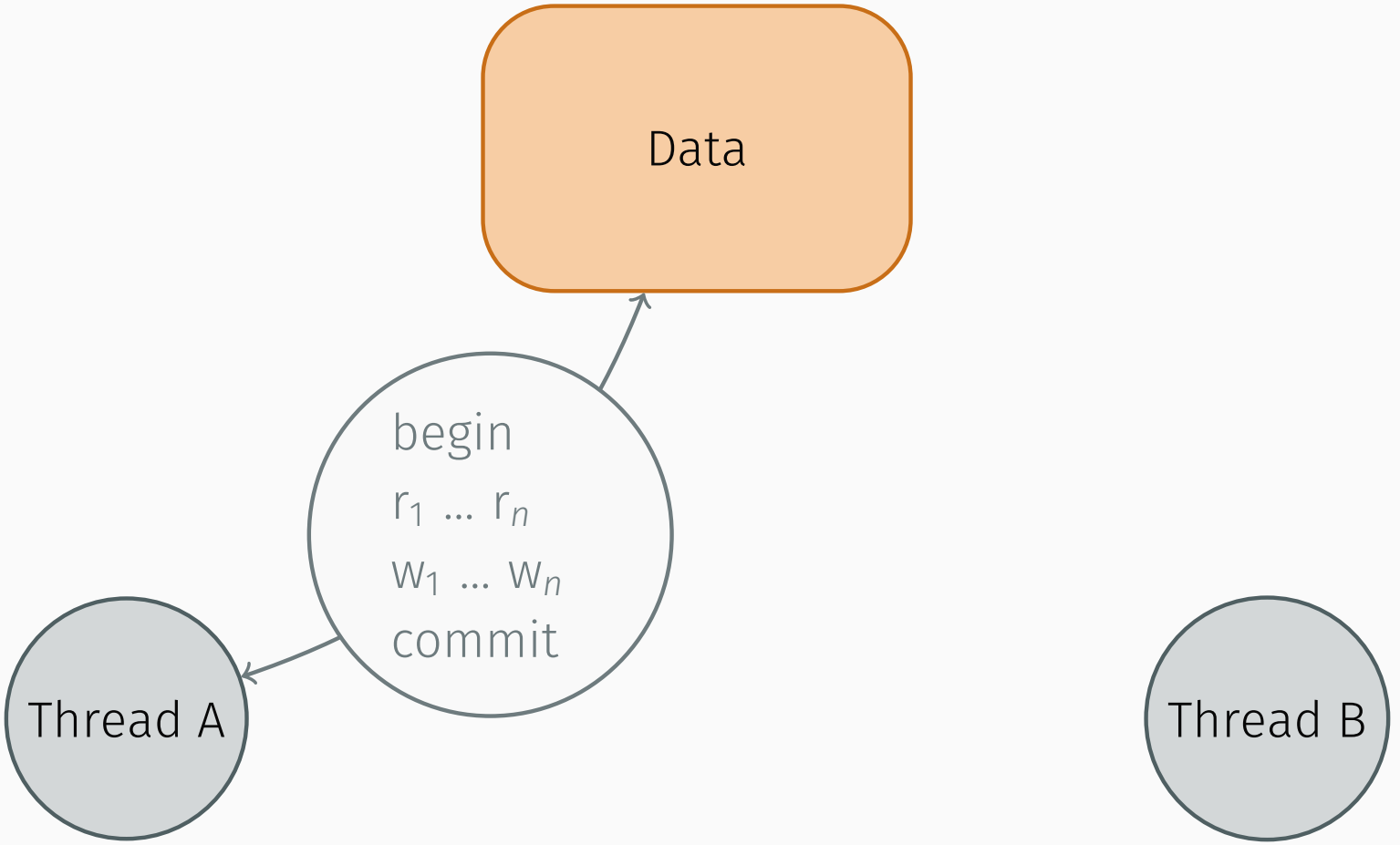


READ-COPY-UPDATE

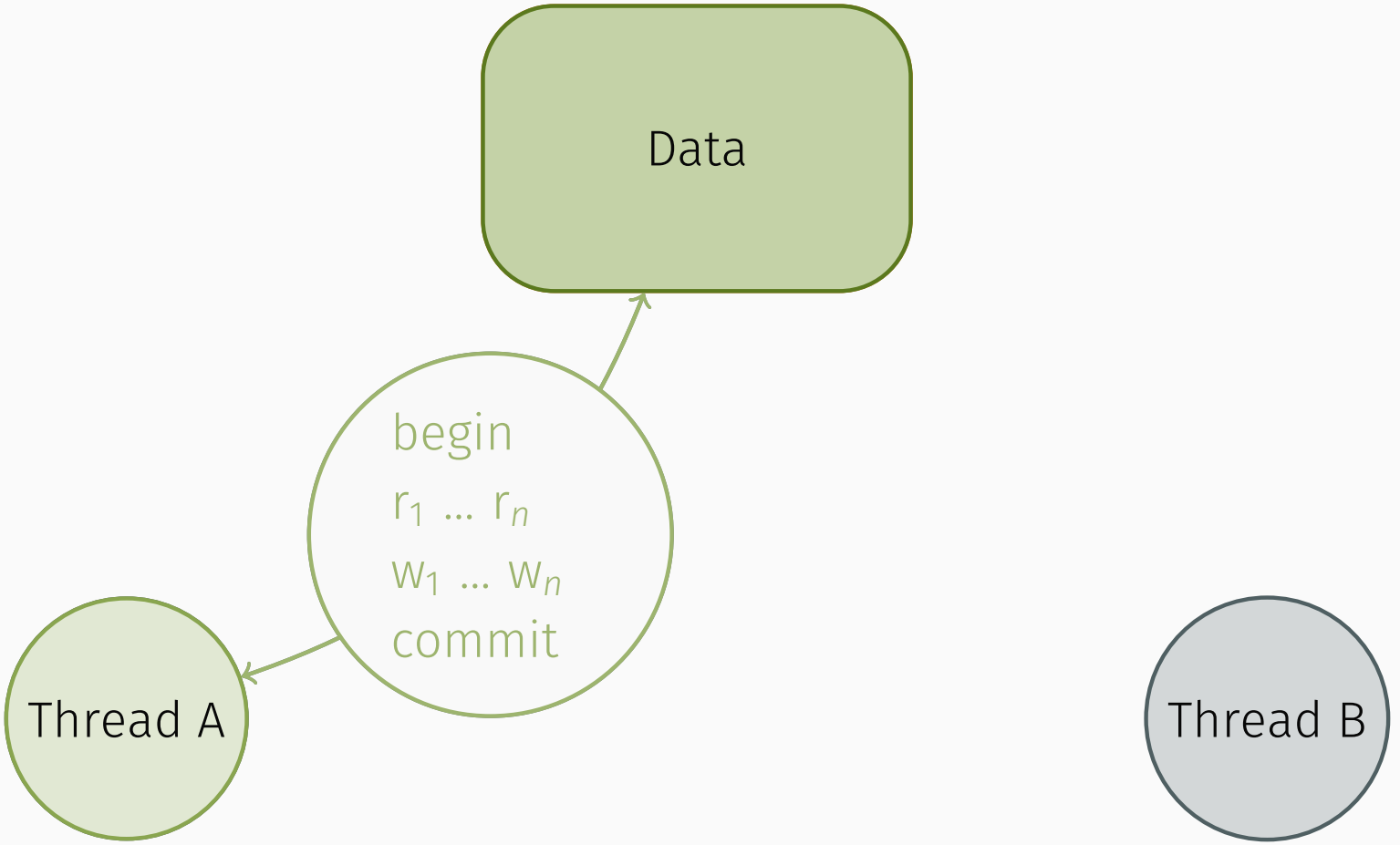


TRANSACTIONAL MEMORY

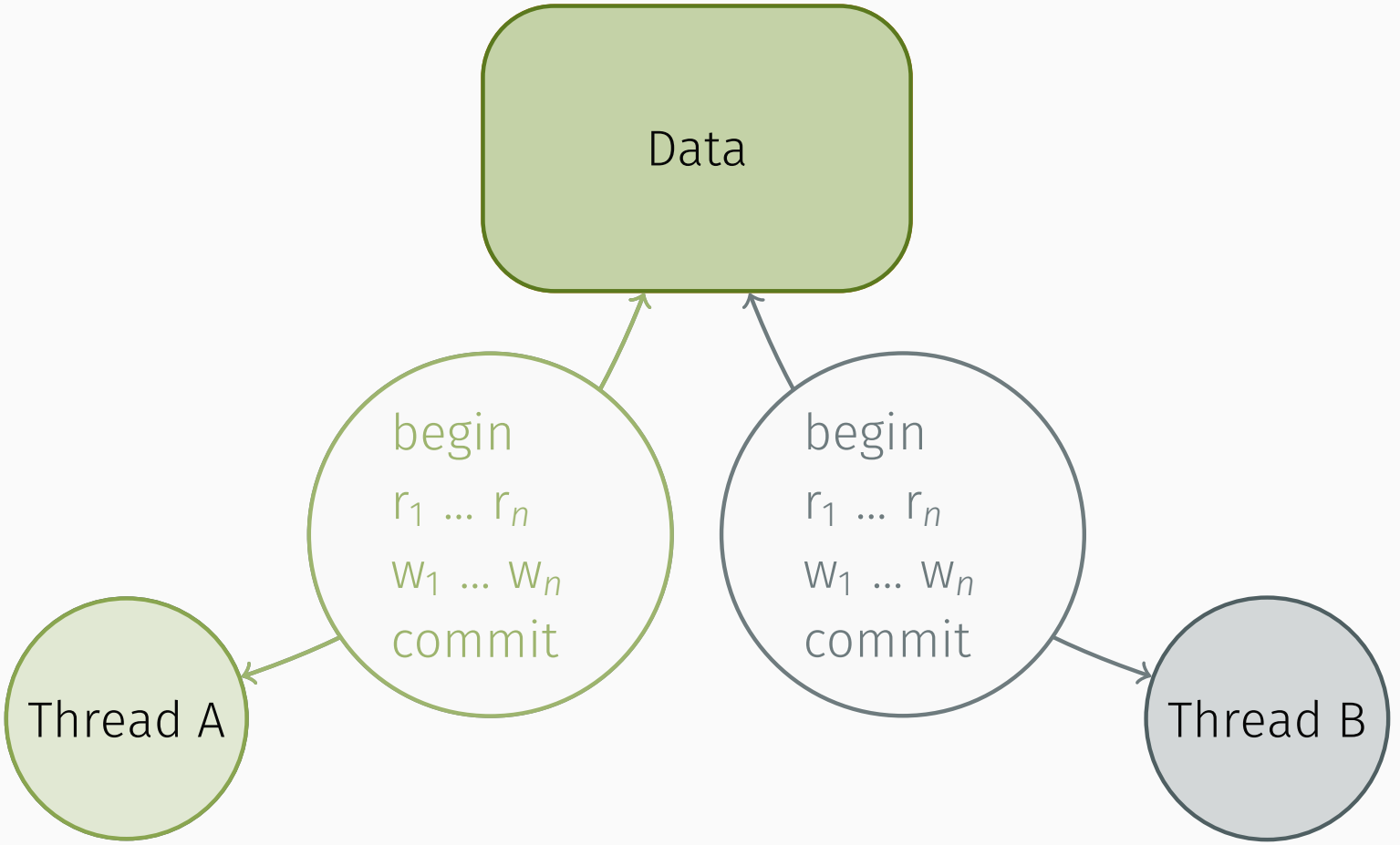
TRANSACTIONAL MEMORY



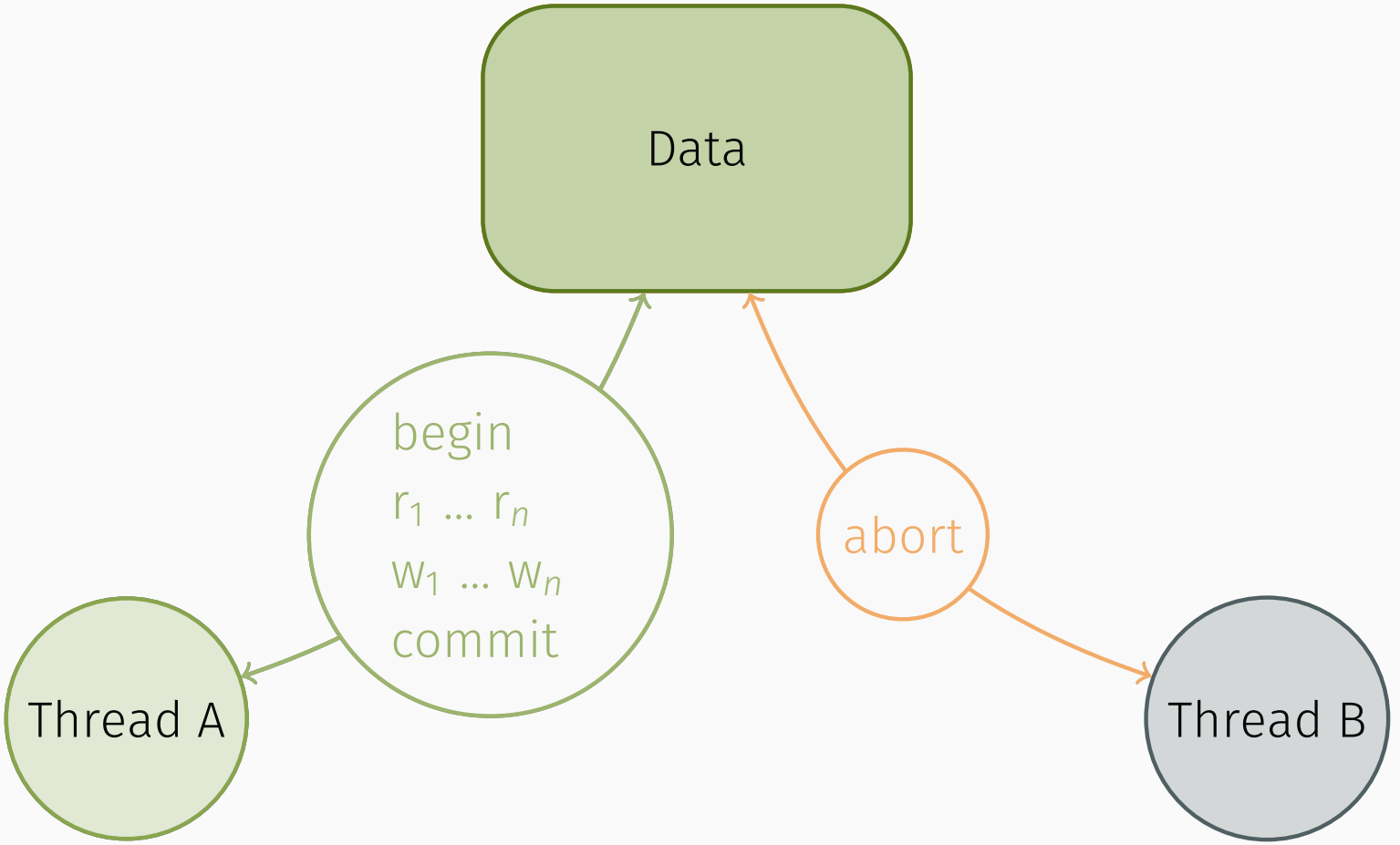
TRANSACTIONAL MEMORY



TRANSACTIONAL MEMORY



TRANSACTIONAL MEMORY



Questions

Which one should be use?

...

Synchrobench

The most extensive comparison of synchronization techniques.

WHAT'S SYNCHROBENCH ?

- Micro-benchmark suite in Java & C/C++
- Open-source [1]
- 31 concurrent data structures
- 5 different synchronization techniques

[1] Available on GitHub https://github.com/gramoli/synchrobench

WHAT'S SYNCHROBENCH ?

- Micro-benchmark suite in Java & C/C++
- Open-source [1]
- 31 concurrent data structures
- 5 different synchronization techniques

8 binary trees
3 dynamic arrays
6 hash tables
6 linked lists
2 queues
6 skip lists

[1] Available on GitHub `https://github.com/gramoli/synchrobench`

WHAT'S SYNCHROBENCH ?

- Micro-benchmark suite in Java & C/C++
- Open-source [1]
- 31 concurrent data structures
- 5 different synchronization techniques

8 binary trees
3 dynamic arrays
6 hash tables
6 linked lists
2 queues
6 skip lists

copy-on-write
mutual exclusion
read-copy-update
read-modify-write
transactional memory

[1] Available on GitHub `https://github.com/gramoli/synchrobench`

WHAT'S SYNCHROBENCH ?

- Micro-benchmark suite in Java & C/C++
- Open-source [1]
- 31 concurrent data structures
- 5 different synchronization techniques

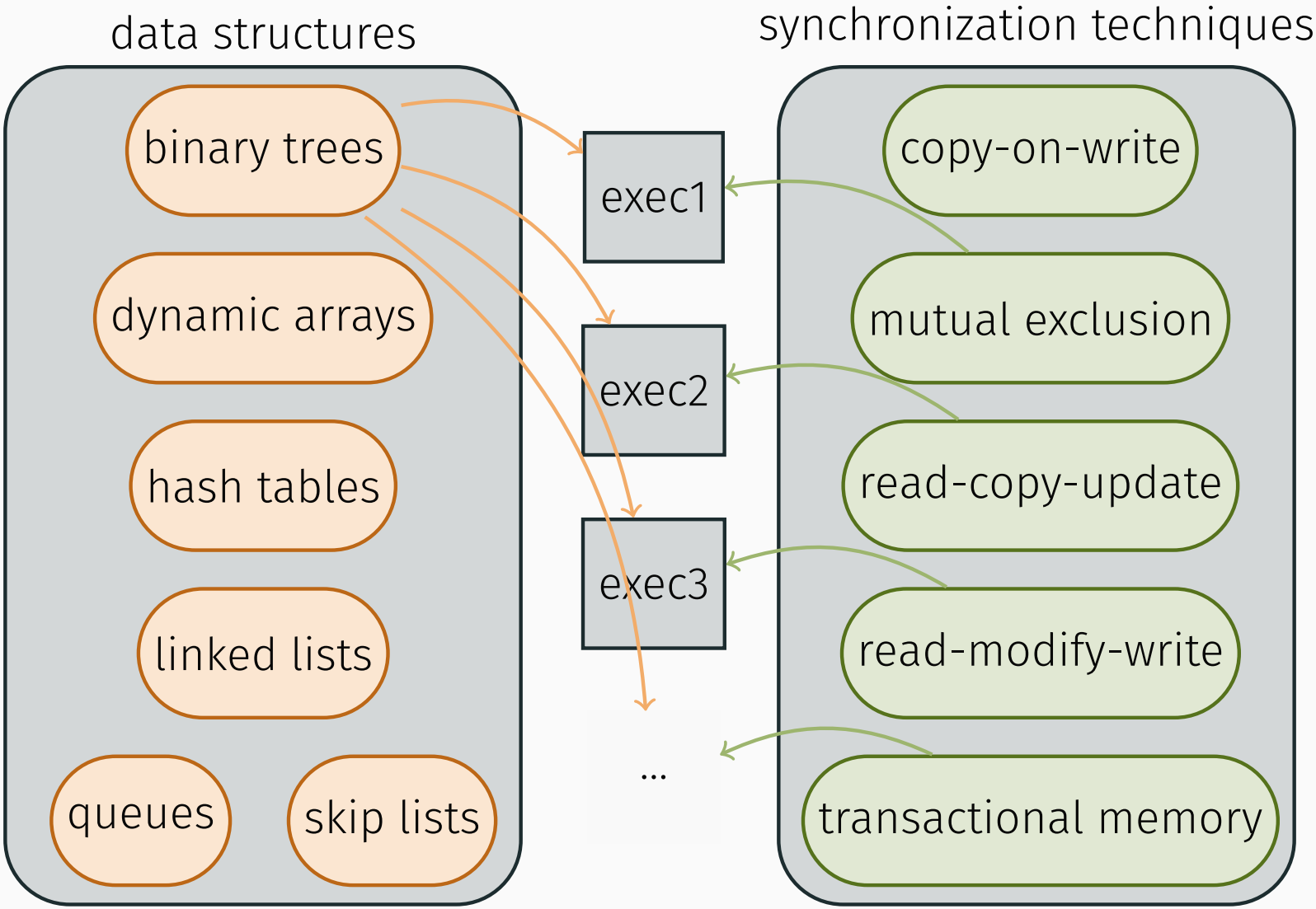
8 binary trees
3 dynamic arrays
6 hash tables
6 linked lists
2 queues
6 skip lists

copy-on-write
mutual exclusion
read-copy-update
read-modify-write
transactional memory

ε-STM [3], SwissTM [5], LSA [7], NOrec [1], PSTM [6], TinySTM [4], TL2 [2]

[1] Available on GitHub `https://github.com/gramoli/synchrobench`

SOME DETAILS



SYNCHROBENCH PARAMETERS

- $t \in \mathbb{N}^*$, no. of threads
- $i \in \mathbb{N}$, initial size
- $r \in \mathbb{N}^*$, value range
- $u \in [0 : 100]$, update ratio
- $f \in \{0, 1\}$, effective update
- $A \in \{0, 1\}$, alternating value
- $d \in \mathbb{N}^*$, duration (milliseconds)
- Java: $W \in \mathbb{N}$, warmup (seconds)
- ...

EXPERIMENTS

Setup

- Average throughput of 5 runs
- Benchmark duration 5 seconds
- Baseline data structure running sequentially (SEQ)

Environment

- AMD Opteron 6378
- **Intel Xeon E5-2450**
- UltraSPARC T2

EXPERIMENTAL RESULTS

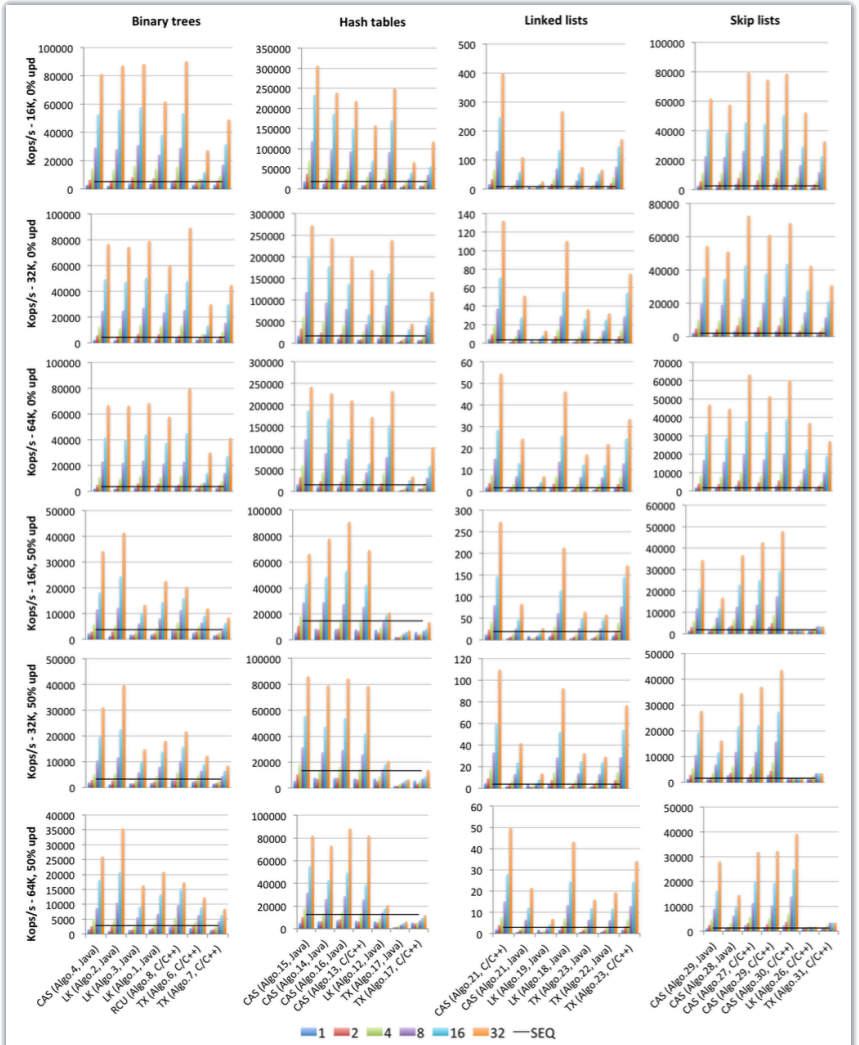
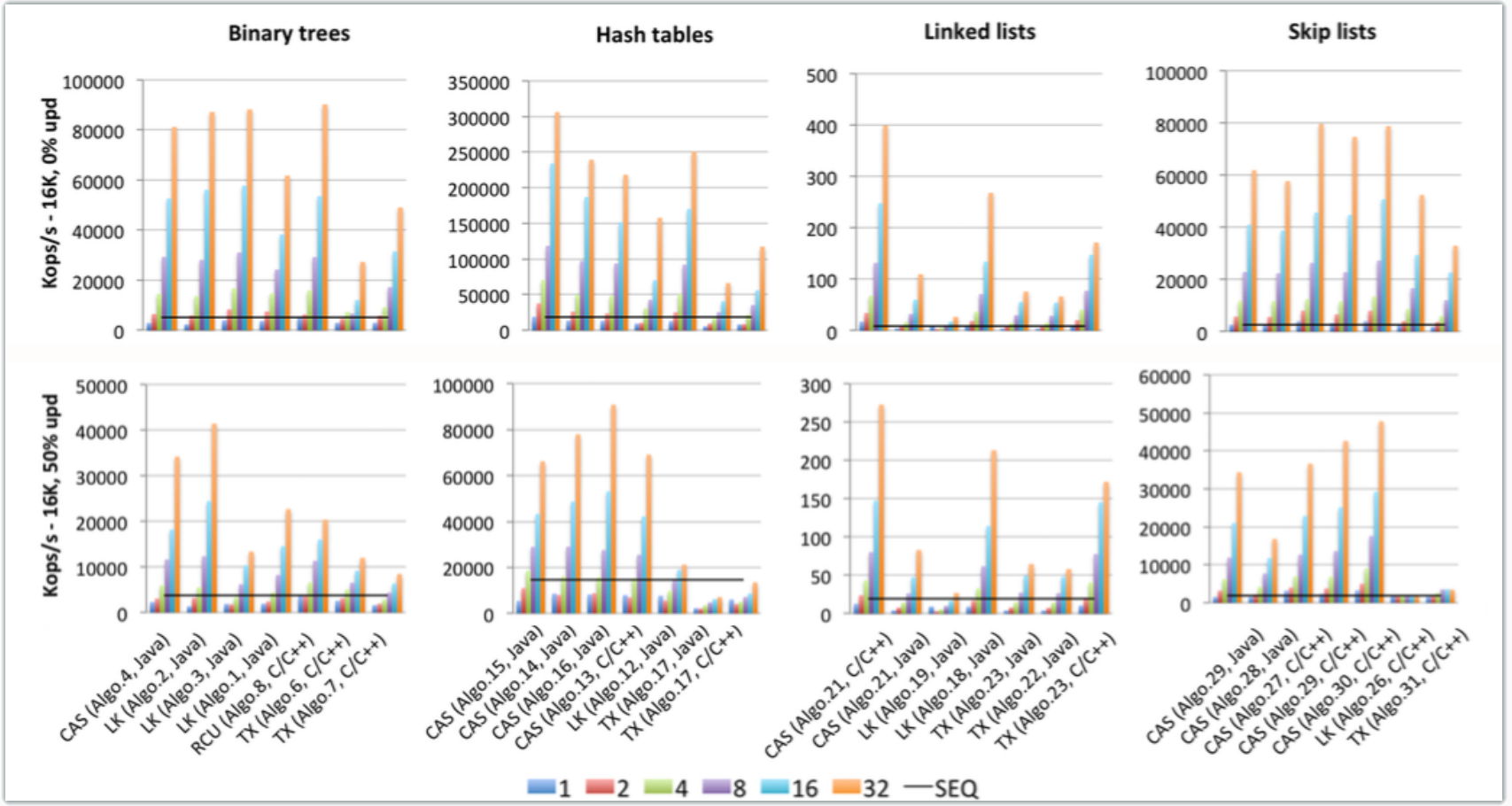


Figure 1. Synchrobench performance results for 1, 2, 4, 8, 16 and 32 threads on the 32-way Intel machine with data structures written in C/C++ and Java, and synchronized with read-modify-write (CAS), locks (LK), read-copy-update (RCU), and transactions (TX), and with sequential performance of non-synchronized data structures as the performance baseline (SEQ)

EXPERIMENTAL RESULTS

CAS ... read-modify-write, LK ... mutual exclusion, RCU ... read-copy-update, TX ... transactional memory



CONCLUSION

Read-modify-write

+ Typically faster than the other techniques
- Design of these data structures difficult

Copy-on-write & Read-copy-update

+ High performance for read intensive workloads
- Low performance for write intensive workloads

Transactional memory (TX) & Mutual exclusion (ME)

· TX offers more consistent performance than ME

REFERENCES

[1] Luke Dalessandro, Michael F Spear, and Michael L Scott. Norec: streamlining stm by abolishing ownership records. In *ACM Sigplan Notices*, volume 45, pages 67–78. ACM, 2010.

[2] Dave Dice, Ori Shalev, and Nir Shavit. Transactional locking ii. In *Distributed Computing*, pages 194–208. Springer, 2006.

[3] Aleksandar Dragojević, Pascal Felber, Vincent Gramoli, and Rachid Guerraoui. Why stm can be more than a research toy. *Communications of the ACM*, 54(4):70–77, 2011.

[4] Pascal Felber, Christof Fetzer, and Torvald Riegel. Dynamic performance tuning of word-based software transactional memory. In *Proc. Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 237–246. ACM, 2008.

[5] Pascal Felber, Vincent Gramoli, and Rachid Guerraoui. Elastic transactions. In *Distributed Computing*, pages 93–107. Springer, 2009.

[6] Vincent Gramoli and Rachid Guerraoui. Reusable concurrent data types. In *ECOOP 2014–Object-Oriented Programming*, pages 182–206. Springer, 2014.

[7] Torvald Riegel, Pascal Felber, and Christof Fetzer. A lazy snapshot algorithm with eager validation. In *Distributed Computing*, pages 284–298. Springer, 2006.

Thank you for your attention.

Q&A