# References

[1] M. Herlihy, "The multicore revolution," in *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, V. Arvind and S. Prasad, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77050-3 pp. 1–8. [Page 1.]

[2] K. Platz, N. Mittal, and S. Venkatesan, "Practical concurrent unrolled linked lists using lazy synchronization," *Journal of Parallel and Distributed Computing*, vol. 139, pp. 110–134, 2020. doi: https://doi.org/10.1016/j.jpdc.2019.11.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0743731519307610 [Pages 1, 2, 3, 5, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 31, 32, 33, 34, 35, 36, 37, 38, 39, 41, 42, 44, 55, 58, and 70.]

[3] M. Herlihy and N. Shavit, *The Art of Multiprocessor Programming, Revised Reprint*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2012. ISBN 9780123973375 [Pages xiii, 1, 3, 7, 8, 9, 10, 11, 12, 13, 14, 15, 36, 38, 44, and 67.]

[4] Z. Shao, J. H. Reppy, and A. W. Appel, "Unrolling lists," in *Proceedings of the 1994 ACM Conference on LISP and Functional Programming*, ser. LFP '94. New York, NY, USA: Association for Computing Machinery, 1994. doi: 10.1145/182409.182453. ISBN 0897916433 p. 185–195. [Online]. Available: https://doi.org/10.1145/182409.182453 [Pages 1 and 17.]

[5] S. Heller, M. Herlihy, V. Luchangco, M. Moir, W. N. Scherer, and N. Shavit, "A lazy concurrent list-based set algorithm," in *Principles of Distributed Systems*, J. H. Anderson, G. Prencipe, and R. Wattenhofer, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. ISBN 978-3-540-36322-4 pp. 3–16. [Pages 2, 3, 5, 13, 17, 18, 19, 23, 26, 32, 35, 38, 67, and 70.]

[6] A. Braginsky and E. Petrank, "Locality-conscious lock-free linked lists," in *Proceedings of the 12th International Conference on Distributed Computing and Networking*, ser. ICDCN'11. Berlin, Heidelberg: Springer-Verlag, 2011. ISBN 364217678X p. 107–118. [Pages 2, 3, and 18.]

[7] K. Platz, N. Mittal, and S. Venkatesan, "Concurrent unrolled skiplist," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, 2019. doi: 10.1109/ICDCS.2019.00157 pp. 1579–1589. [Pages 3, 18, 36, and 59.]

[8] V. Gramoli, "More than you ever wanted to know about synchronization: Synchrobench, measuring the impact of the synchronization on concurrent algorithms," in *Proceedings of the 20th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, ser. PPoPP 2015. New York, NY, USA: Association for Computing Machinery, 2015. doi: 10.1145/2688500.2688501. ISBN 9781450332057 p. 1–10. [Online]. Available: https://doi.org/10.1145/2688500.2688501 [Pages xiii, 13, 19, 22, 25, 28, 32, 38, 39, 58, and 67.]

[9] M. M. Michael, "High performance dynamic lock-free hash tables and list-based sets," in *Proceedings of the Fourteenth Annual ACM Symposium on Parallel Algorithms and Architectures*, ser. SPAA '02. New York, NY, USA: Association for Computing Machinery, 2002. doi: 10.1145/564870.564881. ISBN 1581135297 p. 73–82. [Online]. Available: https://doi.org/10.1145/564870.564881 [Pages 14, 15, 17, and 38.]

[10] T. L. Harris, "A pragmatic implementation of non-blocking linked-lists," in *Distributed Computing*, J. Welch, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. ISBN 978-3-540-45414-4 pp. 300–314. [Pages 14, 17, and 38.]

[11] R. Bayer and E. McCreight, "Organization and maintenance of large ordered indices," in *Proceedings of the 1970 ACM SIGFIDET (Now SIGMOD) Workshop on Data Description, Access and Control*, ser. SIGFIDET '70. New York, NY, USA: Association for Computing Machinery, 1970. doi: 10.1145/1734663.1734671. ISBN 9781450379410 p. 107–141. [Online]. Available: https://doi.org/10.1145/1734663.1734671 [Page 17.]

[12] V. Gramoli, P. Kuznetsov, S. Ravi, and D. Shang, "A concurrency-optimal list-based set," *CoRR*, vol. abs/1502.01633, 2015. [Online]. Available: http://arxiv.org/abs/1502.01633 [Pages 18, 35, 36, 37, 38, 39, 45, and 58.]

[13] L. Frias, J. Petit, and S. Roura, "Lists revisited: Cache-conscious stl lists," *ACM J. Exp. Algorithmics*, vol. 14, Jan. 2010. doi: 10.1145/1498698.1564505. [Online]. Available: https://doi.org/10.1145/1498698.1564505 [Page 18.]

[14] Oracle Corporation, "ConcurrentSkipListSet (Java Platform SE 8)," Jul 2021, last accessed on 2021-08-11. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentSkipListSet.html [Page 40.]

[15] M. Spiegel, *Cache-conscious concurrent data structures*. University of Virginia, 2011. [Page 40.]

[16] Oracle Corporation, "ConcurrentHashMap.KeySetView (Java Platform SE 8)," Jul 2021, last accessed on 2021-08-11. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.KeySetView.html [Pages 40 and 53.]

[17] ——, "ConcurrentHashMap (Java Platform SE 8)," Jul 2021, last accessed on 2021-08-11. [Online]. Available: https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ConcurrentHashMap.html [Page 40.]

[18] M. Naftalin and P. Wadler, *Java Generics and Collections*. O'Reilly Media, Inc., 2006. ISBN 0596527756 [Page 53.]

# Appendix A

# Source Code

## A.1 Lazy List

The Java implementation of the lazy list provided here is based on the implementation from the Synchrobench micro-benchmark suite [8], which itself is based on the implementation from Herlihy and Shavit's book *The Art of Multiprocessor Programming* [3]. The original algorithm was designed by Heller et al. [5].

Listing A.1: Lazy List Source Code

```
1  public class Node {
2    public final int key;
3    public volatile Node next;
4    public volatile boolean marked;
5    private final Lock lock;
6
7    public Node(final int key) {
8      this.key = key;
9      this.lock = new ReentrantLock();
10     marked = false;
11   }
12
13   public void lock() {
14     this.lock.lock();
15   }
16
17   public void unlock() {
18     this.lock.unlock();
19   }
```