Lock-Free Programming: Threads Share Data Without Waiting on a Lock

Kabui, Charles

2026-08-25

Read at ToKnow.ai

Lock-free programming lets several threads share data without making them take turns on a mutex, the lock that normally pauses one thread while another holds it. Threads coordinate with atomic operations instead, most often compare-and-swap, a single indivisible step that updates a value only if it still matches an expected one. Because no thread ever blocks, a slow or paused thread can't stall everyone else. The idea traces to Herlihy's 1991 wait-free

synchronization paper, and the most used structure, a lock-free first-in-first-out queue, comes from Michael and Scott's 1996 paper. The hard part isn't the atomic step; it's reclaiming memory safely and beating the ABA problem, where a value changes and then changes back.

Locks get expensive exactly where they're needed most. Under heavy contention, waiting on a lock means kernel arbitration and cache misses that can cost more than the work itself. At LMAX, a financial exchange, queues between processing stages added as much latency as disk I/O, hundreds of microseconds per hop. Its lock-free Disruptor cut mean latency for a three-stage pipeline by three orders of magnitude, handled roughly eight times the throughput, and passed over 25 million messages per second at under 50 nanoseconds. The same priorities drive NautilusTrader, an open-source Rust trading engine with a deterministic, event-driven architecture built for low-jitter execution.

None of this makes locks obsolete. Lock-free code is famously hard to reason about, wait-free structures are rarer still, and most production code is lock-free only with careful memory reclamation on top. But when mutexes become the bottleneck, there is a proven ladder above them: atomic operations, then lock-free queues and rings, then wait-free designs.

Read More: Bloom filters, another elegant trick that powers systems at scale

Sources:

- Herlihy (1991): Wait-Free Synchronization (PDF)
- Michael & Scott (1996): Simple, Fast, and Practical Non-Blocking and Blocking Concurrent Queue Algorithms (PDF)
- LMAX Disruptor: High Performance Alternative to Bounded Queues (Technical Paper)
- Preshing: An Introduction to Lock-Free Programming
- NautilusTrader: Rust-Native Trading Engine (GitHub)

Disclaimer: *For information only. Accuracy or completeness not guaranteed. Illegal use prohibited. Not professional advice or solicitation.* ***Read more:*** */terms-of-service*