MLFQ Scheduler - Week 3

Muhammad Hussain - 29004
Sarfaraz Ahmed - 24520

December 4, 2025

Overview

This week extends the xv6-RISC-V MLFQ scheduler with two new system calls, a complete 7-test validation suite, and real-time monitoring utilities. The implementation verifies correct CPU-bound demotion, I/O fairness, mixed workload handling, and starvation prevention.

System Calls

1. **boostproc():** Allows manual boosting of either a specific process or all processes to the highest-priority queue (Q0). Used for testing, debugging, and controlled priority recovery.
2. **mlfq_stats():** Returns real-time statistics including total schedules, boosts, demotions, per-queue counts, and scheduling frequency. A spinlock ensures atomic data snapshots.

Starvation Prevention

MLFQ prevents starvation through periodic priority boosting. Every fixed interval (100 ticks), all processes are moved back to Q0 regardless of their previous demotions. This ensures:

- Long-running CPU-bound tasks eventually regain high priority.
- Low-priority queues do not block higher wait-time processes.
- All queues participate in scheduling over time.

This mechanism guarantees fairness and prevents starvation, even when CPU-bound tasks continuously consume full quanta.

Testing Summary

A seven-test suite validates scheduler correctness:

- **CPU-Bound Demotion:** Verifies Q0→Q1→Q2→Q3 transitions.
- **I/O Fairness:** Confirms I/O-bound tasks remain in Q0.
- **Mixed Workload:** Demonstrates fair coexistence of CPU and I/O tasks.
- **Auto Priority Boost:** Ensures global boosts occur at fixed intervals.
- **Manual Boost (Single + All Processes):** Verifies correct functioning of boostproc().
- **Starvation Test:** Long-running CPU tasks repeatedly demote and boost, proving starvation prevention.

Real-Time Monitoring

A user-level tool displays live scheduler behavior, including:

- Queue population
- Scheduling frequency per queue
- Total boosts and demotions

This assists in visually confirming fairness, boost events, and workload signatures.

Conclusion

The MLFQ scheduler is fully functional with priority boosting, kernel statistics, and a comprehensive test suite. All components work together to ensure fairness, prevent starvation, and provide measurable insight into real scheduler dynamics, suitable for analysis and viva demonstrations.