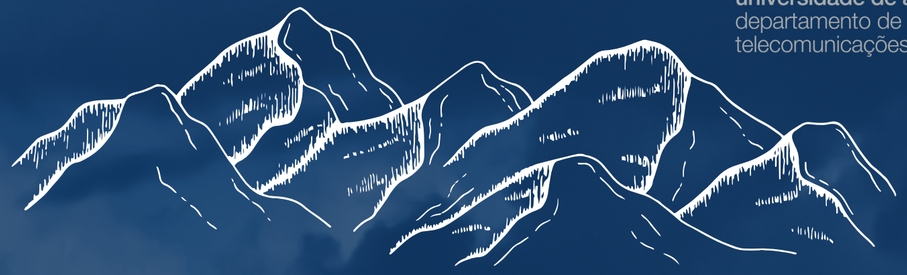
deti
universidade de aveiro
departamento de eletrónica, telecomunicações e informática



WEATHER STATIONS

MULTI-THREADED PIPELINE

Carolina Reis | nº 131193
Eduardo Lopes | nº 103070

OBJECTIVES

1

Synchronization

Explain the synchronization mechanisms used in the implementation.

2

Safety

Show how deadlocks and starvation are prevented by design.

3

Performance

Present benchmark evidence and practical tuning decisions.

ARCHITECTURE OVERVIEW

MAIN PRODUCER/CONSUMER PIPELINE

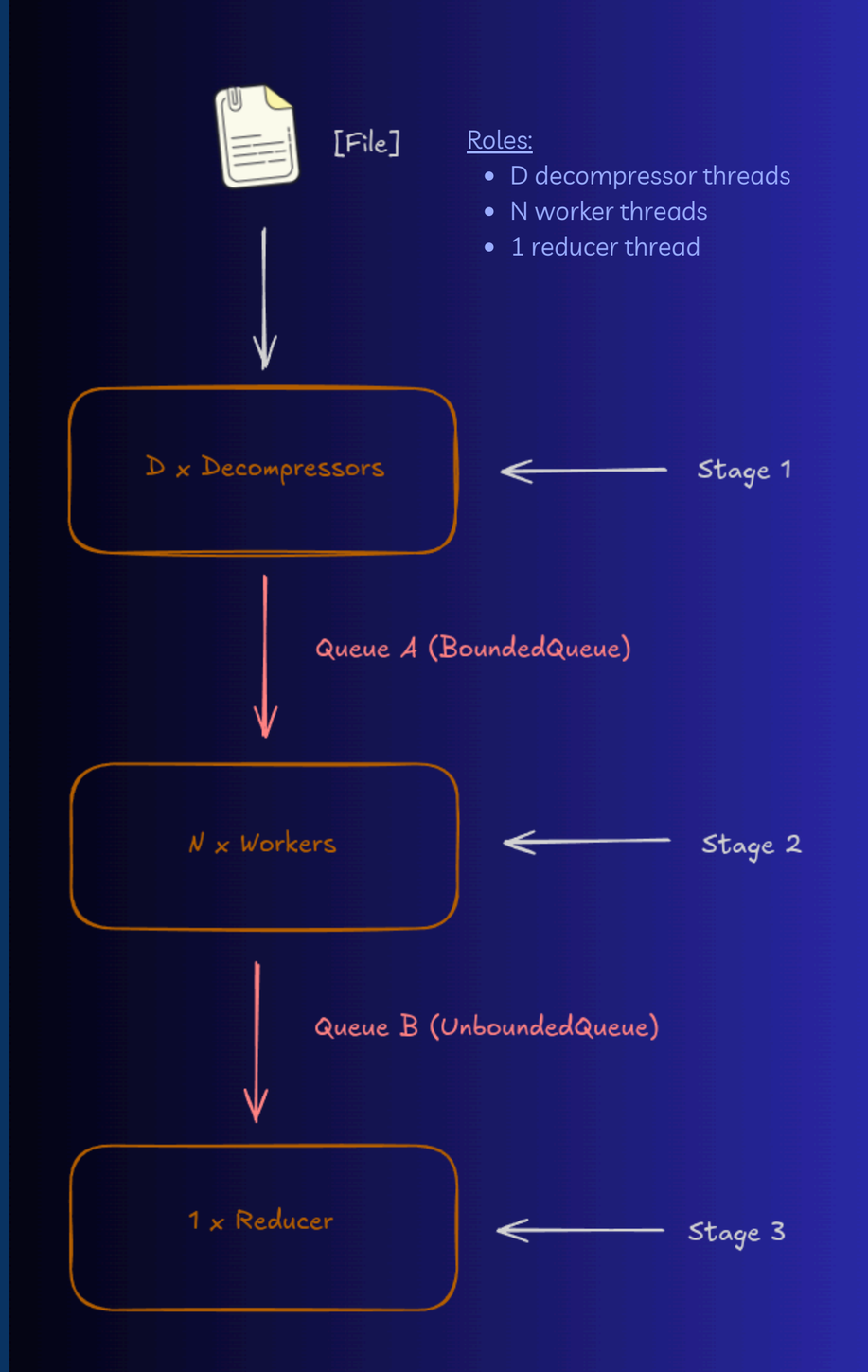
Pre-scan & I/0:

- Build vector<BlockIndex> once at startup.
- Decompressors claim work with fetch_add.
- Each read uses pread(offset).

Flow control & memory:

- Queue A is bounded (4..8) for backpressure.
- Queue B is unbounded for lightweight merge state.
- Buffer pool is bounded: queue_a_cap + 4.

Design principle: stages communicate only through queues; no shared mutable global state in the hot path.



SYNCHRONIZATION MECHANISMS

Used in implementation:

- **Atomic block dispatch:**
  next_block_.fetch_add(1)
- **Parallel file I/O:**
  pread() with precomputed offsets
- **Queue A:**
  bounded queue with not_full + not_empty
- **Queue B:**
  unbounded queue with not_empty
- **Pool queue:**
  bounded queue for buffer reuse + memory cap

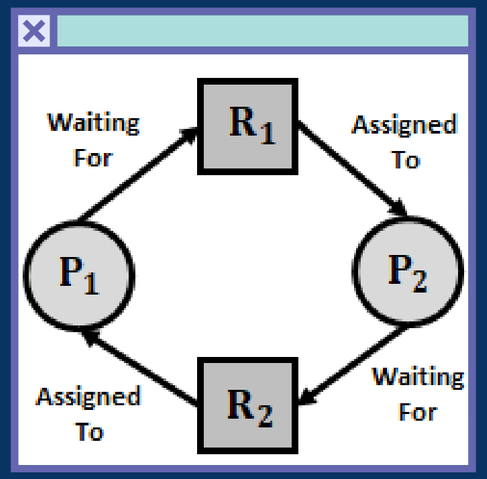
Why this matters:

- No shared file pointer lock
- Backpressure where neeeded (Queue A)
- Controlled memory footprint via pool
- Simple queue contracts reduce concurrency bugs

DEADLOCK PREVENTION STRATEGY

NO CIRCULAR WAIT

- No nested lock ownership across queue resources.
- Queue operations are single-resource critical sections.
- Worker lock sequence is separated in time: pop Queue A, push Queue B, push Pool.



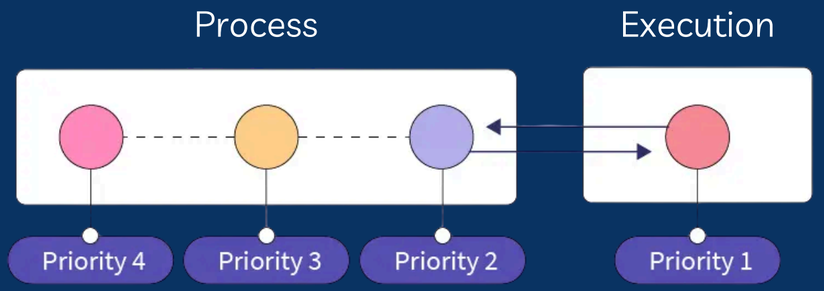
TERMINATION CASCADE (ORDERED SHUTDOWN)

close() performs **notify_all()**, so blocked waiters wake and exit.

STARVATION PREVENTION STRATEGY

PROGRESS GUARANTEES

- Atomic dispatch gives each block a unique owner.
- Queue waits always use predicates.
- notify_one() on normal flow keeps forward progress.
- notify_all() on close releases every blocked thread.



Queue A rationale

Bounded queue applies backpressure when workers are slower.

Queue B rationale

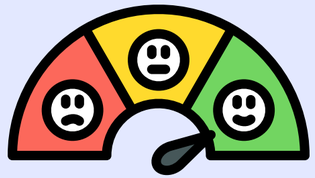
Unbounded queue avoids unnecessary producer blocking; reducer is merge-only.

PERFORMANCE SNAPSHOT

Blocks	Single-thread	D=2, N=4	D=2, N=8	D=2, N=14	D=4, N=4	D=4, N=8	D=4, N=14	Speedup
10	13253 ms	4711 ms	3938 ms	4469 ms	4927 ms	3973 ms	3938 ms	3.37x
25	33916 ms	11198 ms	10025 ms	9914 ms	10781 ms	8437 ms	8297 ms	4.09x
50	70034 ms	19323 ms	17567 ms	17536 ms	19177 ms	14393 ms	13766 ms	5.09x
100	143108 ms	36380 ms	29289 ms	29549 ms	36855 ms	25833	24690 ms	5.80x

Interpretation:

- Dominant bottleneck is block production (I/O + decompression).
- Increasing D helps more than increasing N after N ≈ 8.
- Practical setup: D=4, N=8..14, Queue A cap=8.



CONCLUSION

MAIN MESSAGE

- We first guarantee correctness and liveness, the optimize throughput by removing serialization from the hot path.

Correctness & Safety

- No deadlocks by design
- No starvation on active flow or shutdown
- Deterministic reducer semantics

Performance

- Lock-free block assignment
- Parallel pread() by offset
- Buffer recycling + bounded backpressure
- Up to 5.80x speedup

DEMO

Live command sequence

```
cd multi_thread && make
./build/cle-samples 10
./run_timed.sh measurements-10.cle 1 1 8
./run_timed.sh measurements-10.cle 4 8 8
```

Analyse elapsed-time delta


THANK
YOU!