

R5 bounded service resources

Each voter admits at most 24 authenticated clients and four incoming handshakes or rejected-client connections within a 32-connection pool. Fixed peers retain reserved capacity. Rejected authenticated requests receive Busy before submission; pending writes that lose their response remain unknown outcomes with a durable retry identity. A client admission limit is not protection against unlimited unauthenticated traffic reaching the operating system.

Client frames are at most 64 KiB, authenticated peer frames at most 2 MiB. An iterative pre-scan caps JSON nesting at 64 before recursive unmarshalling. Protocol objects need much less depth; SQL strings containing braces or escaped quotes do not count as nested objects. Each connection holds one incoming frame and at most 256 outgoing frames, additionally limited to 2 MiB total. One bounded frame per client connection per event-loop turn prevents one pipelined reader from monopolizing dispatch. Under SOD 0005, authenticated peer input fills a shared 128-packet turn buffer, stepped in batches of at most sixteen; partial frames and snapshot controls stop a receive burst. Peer output has at most 256 flush attempts per turn, stopping when TLS would block. These replace the earlier 64-frame queue and eight-frame peer bursts; the additional inline packet storage is bounded but must be included in RSS accounting. Slow peer traffic is dropped for retransmission rather than accumulated without limit; catch-up traffic has its own bounded transfer buffers and activity timeout. Kernel TCP buffers are separate operating-system resources, with the number of sockets bounded by the connection pool.

The consensus memory window is 64 slots and incoming transition batches are bounded. Session metadata has 65,536 rows; explicit replicated retirement advances the epoch before reclaiming rows. A node runs one snapshot/generation maintenance job. Chunked copy/transfer, cooperative cancellation and SQLite progress callbacks bound individual worker turns; cancellation cannot interrupt a blocked kernel storage call. Generation/image ownership inventories and the 8 GiB aggregate consensus-history quota protect active predecessors and transition reserves. Application data capacity is separate from retained consensus history.

Read results are bounded by both 4,096 rows and 256 KiB of accounted result storage, with a separate one-million-instruction approximate SQLite progress budget. JSON wire responses have their own size bound. Expensive reads fail without a partial result. Preview work rolls back and may use a local work budget. Replicated writes do not convert timing or resource failures into SQL rejections: unknown execution, allocation and storage failures stop acknowledgement and require repair/replay. R1.1 owns the complete supported SQL execution contract. Expensive permitted SQL consumes CPU while executing; this document does not promise constant latency for arbitrary user SQL or resilience when every voter exhausts its resources.

Measurement and fault evidence

`tools/check_resource_load.py` runs 60 active clients, 12 slow readers, 12 excess client attempts and stalled handshakes on three disk-backed processes. It records per-voter CPU time, RSS and kernel I/O counters while requests include fresh reads, successful writes, instruction-limited reads, a costly permitted aggregate write, and live snapshot/compaction. Every acknowledged counter update is verified and the costly write agrees on all voters. The slow readers pipeline responses without consuming them, exercising queue backpressure rather than only idle connections.

`SQLODIN_RESOURCE_PROFILE` is disabled in ordinary builds. Qualification builds use Odin's synchronized allocation trackers for the service heap and main-thread temporary allocator, count explicit frame-copy bytes, and record SQLite's own global heap peaks. These counters measure their stated allocation/copy paths; they do not claim to count every TLS/libc copy, kernel buffer or virtual

mapping. Tracking-map overhead changes timing, so instrumented throughput is not a release performance result. Ordinary-build CPU/RSS measurements remain separate.

The corrected `resource-load-slow-readers-linux-v3.json` records 3,840 operations at full client admission and twelve rejected excess connections. Before leaving responses unread, each slow connection validates a 120,000-byte result; pipelined requests include the required timeout. Peak ordinary voter RSS is 29–35 MiB and later sampling quarters remain below the initial peak. Every acknowledged update agrees, the expensive permitted write completes, and all voters publish the live snapshot generation. Earlier v1/v2 workloads accidentally omitted that timeout and returned small errors; they are retained but do not prove large-response backpressure. The earlier zero-allocation profile was an instrumentation failure.

`resource-profile-slow-readers-linux-v3.json` records about 37–41 thousand tracked heap allocations per voter, 15.1 MB peak tracked heap, 15.2 MB SQLite heap peak, and about 20 MB explicit frame copying over the bounded campaign. Temporary allocator peaks range from 0.53 to 1.03 MB. Repeated allocation and wire encoding are measurable R6 optimization inputs, not evidence of a growing live heap. The short ordinary run's idle samples consume at most about one percent of a core. Full admission and excess-client rejection establish bounded overload behavior; these samples are not maximum-throughput or large-dataset claims.

Storage fault injection targets exact consensus and application WAL paths after successful seed writes. ENOSPC, actual partial writes followed by EIO, and sync EIO must stop the failed voter without acknowledging the affected request. Survivors resolve the same identity once, then the repaired voter catches up and all voters restart with acknowledged data intact. This is controlled syscall-failure testing, not physical power-loss certification. Separate crash matrices cover journal, application, backup, manifest, generation publication and retirement boundaries; corruption regressions reject invalid identities, descriptors, checksums and images. R4 supplies the supported repair, restore and fenced-replacement procedures.

Capacity verification budgets

The 100 GiB / 4 KiB-row profile contains 26,214,400 application rows before metadata. The earlier ten-million-row verifier limit could not cover that planned workload regardless of available storage. R6 increases the local worker budget to 32 million rows, 4 GiB of disposable disk scratch, eight billion VM instructions and 3,600 seconds per copy/hash or verification pass; the image limit remains 128 GiB. The scratch database keeps its 2 MiB page cache, and the source reader keeps its 8 MiB cache. Neither row count nor scratch capacity becomes an in-memory allocation. These are maximum worker budgets, not required test durations or service request deadlines. Cancellation still applies at copy/hash/VM boundaries.

Snapshot capture, generation copying, offline migration, backup hashing/copying and restore copying share the same 3,600-second maintenance ceiling. Backup verification uses the same eight-billion-instruction ceiling. These local limits do not change the five-second quorum-recovery or sixty-second restart targets, nor allow publication of an incomplete or unverified image.

The logical digest encoding, sorted row hashes, receipt/certificate identity and publication rules are unchanged. Verification that exceeds a budget fails without publishing a receipt. Disk exhaustion remains fail closed; operators need space for active, retained and staged images plus scratch. Measured dataset and physical allocation results must identify the build and actual capacity reached; larger budgets alone do not establish a successful capacity qualification.

Live generation replay scheduling

Generation catch-up copies at most 128 journal records / 1 MiB per owner turn, then applies at most one SQL transaction before returning to client/peer service. A small INSERT SELECT record can expand into a large data write; a record-count bound alone must not batch thirty-two such writes onto one uninterrupted turn. The replacement remains private until its journal, application prefix, promises and identity agree with the live owner and catalog publication completes. This scheduling change preserves that publication rule and the existing physical crash boundaries. It cannot preempt one SQLite call or a blocked kernel I/O call. The generation regression checks the one-transaction bound alongside 140 tail writes, an unchosen vote, a global promise, reserved IDs and durable restart.

The read VM callback budget is not a wall-clock or disk-byte bound. SQLite's optimized unqualified count can traverse a large B-tree inside one VM opcode. A long statement still occupies the serialized service owner; kernel I/O cannot be preempted by this scheduler. Capacity verification therefore checks complete row coverage with bounded primary-key ranges, rather than treating a whole-table count as a short transactional query. The original failed whole-table scan and its recovery timing remain evidence. This does not establish an analytical-query latency guarantee or change the short-query workload targets.