

Simutrans — freelist_iter_tpl skip-empty

Performance analysis (revalidated)

Trunk be2c53104 (r12100) · 2026-07-23 · category A (VALID) · not committed, not posted

Historical correction (read first)

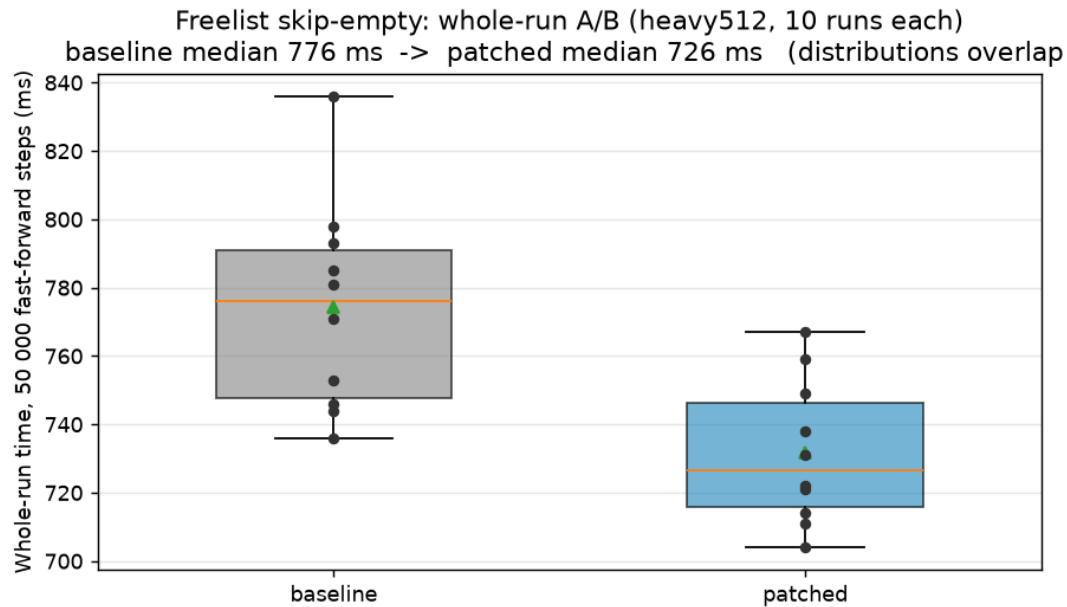
An earlier version of this analysis presented a subsystem split in which `private_car` movement was about 50% of the simulation step, measured with a per-phase cycle timer. That figure was a measurement artifact and is withdrawn. The benchmark save had essentially zero live city cars; the “cycles” were a few virtual-machine wall-clock stalls miscounted by the TSC. This document contains no `private_car` subsystem split, no per-phase `private_car` cycle numbers, and no projection built on that 50%. The withdrawal does not by itself invalidate the freelist patch, whose evidence never depended on that split. No `private_car` patch is proposed.

1. Executive summary

`freelist_iter_tpl<T>::sync_step()` iterates its per-type object pool by testing every slot of every chunk, including free ones. In the sparse, high-churn pools (smoke, wildlife, road signs, and partly pedestrians) most slots are empty most of the time, so most of that per-step work is wasted `test(i)` calls on empty slots.

The patch replaces the scan-every-slot loop with a next-set-bit scan over the occupancy bitmap: it jumps from one live slot to the next. Same objects, same ascending order, same behaviour — only the empty-slot testing is removed. `std::bitset` (no portable next-set-bit) is replaced by a small POD bitmap of the same bit count with a portable trailing-zero scan. One file, +72/−15 lines.

Re-measured on current trunk (heavy512, pak128.german, 50,000 fast-forward steps, 10 interleaved runs per build): baseline median 776 ms → patched median 726 ms. Every one of the 10 interleaved pairs was faster patched (10/10), mean gap ~43 ms (~5–6%). The marginal distributions overlap, so this is reported as a modest win carried by the paired design and the mechanism, not as a non-overlapping 8%.



2. The structural problem

freelist_iter_tpl<T> is a static per-type memory pool (users: private_car_t, pedestrian_t, movingobj_t, wolke_t, pumpe_t, senke_t). Each chunk holds an occupancy bitmap plus new_chunk_size fixed slots. The sync walk tested every slot of every chunk. In sparse pools the bitmap is mostly zero between the peaks that allocated it, so the walk spends most of its time testing empty slots.

occupancy bitmap of one chunk (illustrative): 7 live of 64 slots. Old loop tests all 64; patch jumps between the 7 set b



3. The patch

The inner loop becomes: for (i = mask.find_first(); i < new_chunk_size; i = mask.find_next(i)). find_first/find_next return the lowest set bit ($\geq$ from) by scanning 64-bit words with a portable count-trailing-zeros. The live bitmap is re-read on each advance, so a just-cleared current bit is

skipped and any in-pass mutation is seen exactly as by the old test(i) walk. The outer chunk-walk and the if(nodecount==0) return early-out are unchanged. The savegame format is not touched.

4. Behaviour equivalence

Standalone test (test_freelist_bitmap.cc, g++ -std=c++17 -O2 -Wall -Wextra, no warnings): for N in {1,63,64,65,127,128,161,200,256} and thousands of random fill patterns, the find_first/find_next sequence is identical to the naive scan-every-index sequence. Edge cases: empty, first slot, last slot, cross-word boundary (63/64), sparse. A mutation test replays self-removal and later-bit-flip decisions verbatim and asserts identical visit sequences. Output: ALL FREELIST BITMAP EQUIVALENCE TESTS PASSED.

Mutation contract verified at trunk for all six users: each sync_step returns only SYNC_OK/SYNC_DELETE; none allocates a same-pool object or clears another object's bit during the pass. In-game invariants identical A/B: route_calls 846–847, final month 23455 in both builds.

5. Portability

uint64/uint32 via simtypes.h; shift amounts always & 63 (0–63); ctz never called with 0 (guarded). Trailing-zero scan: GCC/Clang __builtin_ctzll; MSVC x64/ARM64 _BitScanForward64; MSVC 32-bit _BitScanForward on both halves. POD, memset-zeroable, ceil(N/64) words, no allocation, no STL-internal APIs, no endianness assumption.

The GCC and Clang paths using __builtin_ctzll were both compiled and exercised by the standalone equivalence harness. The MSVC _BitScanForward64 and 32-bit fallback paths were reviewed statically but were not compiled in this environment. Compilers: GCC 16.1.0 and Clang 22.1.8 (both x86_64-w64-windows-gnu, 64-bit), each -std=c++17 -O2 -Wall -Wextra -Wpedantic, zero warnings, both runs ALL FREELIST BITMAP EQUIVALENCE TESTS PASSED with identical output.

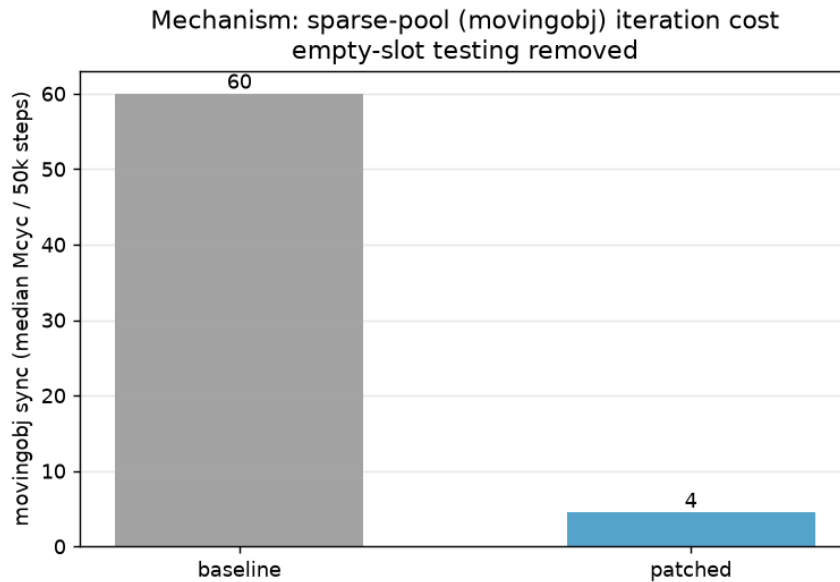
6. Whole-run benchmark

Environment: Shadow PC virtual machine, AMD EPYC 7543P, 8 logical CPUs; Windows 11; g++ 16.1.0; RelWithDebInfo -O2 -g -DNDEBUG; headless (backend none). Baseline and patched share the same commit, scenario, build mode and command line, differing only by the patch. Warm-ups discarded; 10 interleaved runs each.

build	n	mean ms	median ms	min	max
baseline	10	774.3	776.0	736	836
patched	10	731.6	726.5	704	767

Paired: 10/10 rounds faster patched (sign test $p \approx 0.001$). Marginal distributions overlap (baseline min $736 \leq$ patched max 767); the effect is carried by the paired design. Mechanism:

the sparse movingobj sync phase dropped from ~60 to ~4 Mcyc (~15×) — the empty-slot waste removed. This is a VM: short windows are noisy, so the headline uses a long window and a paired comparison, and per-phase cycle sums are used only as mechanism hints, never as magnitudes.



7. Status against current trunk

Trunk be2c53104 (r12100) is unchanged since the patch was authored: the header still uses `std::bitset` and the scan-every-slot loop, so upstream has not absorbed an equivalent change. `git apply --check` is clean on a pristine r12100 tree; both builds compile with no new warning (65 warnings each, none in `freelist_iter_tpl.h`, identical A/B warning sets).

8. The `private_car` correction (explicit)

The earlier “`private_car` $\approx$ 50% of the step” figure was a VM timer artifact and is withdrawn (full audit: `privatecar-timer-audit/`). This alone does not invalidate the freelist patch: the freelist evidence is the standalone equivalence test, the whole-run A/B, and the `route_calls/month` invariants — none of which used the `private_car` cycle number. The freelist patch never claimed to speed up `private_car`; that pool is dense, so skipping empty slots cannot help it, and the win is in the sparse pools. No `private_car` code, instrumentation or behaviour is modified, and the `private_car` investigation is not reopened.

9. Evidence that remains valid

- Standalone bitmap-equivalence test: order and selection identical across sizes, edge cases and mutation.

- Whole-run A/B, 10 interleaved runs, paired 10/10 faster patched (~5–6%).
- Mechanism: sparse-pool (movingobj) sync ~15× cheaper.
- In-game equivalence invariants identical (route_calls, final month).
- Applies clean to current trunk; compiles with no new warning.

10. Limitations

- One machine (a VM), one pakset (pak128.german), one workload, fast-forward only. Windowed FPS gain is smaller.
- Whole-run marginal distributions overlap; the claim relies on the paired design and the mechanism, stated as ~5–6%.
- GCC and Clang (64-bit) both compiled and ran the equivalence harness; MSVC _BitScanForward64 and the 32-bit fallback were reviewed statically but not compiled in this environment.
- The earlier “89% empty-slot” in-game counter was not re-wired (to keep the shipped header pristine); the mechanism is evidenced by the movingobj ~15× drop instead.

11. Decision

Category A — VALID. Equivalence demonstrated, applies clean to current trunk, improvement reproduced (modest, ~5–6%, paired 10/10), complexity proportionate, portable by construction. Delivered for review; not committed, not posted.