High-Performance Smith–Waterman Implementation

Assumptions, Compiler Flags, and Optimization Journey

2024JCS2045 2024JCS2612

Abstract

This document provides a complete description of the assumptions, hardware requirements, compiler flags, and iterative optimization process used to develop the final high-performance Smith–Waterman (SW) implementation. The final system supports hybrid 16-bit and 32-bit SIMD execution, anti-diagonal wavefront parallelism, AVX2-accelerated prefix scans, and cache-blocked tiling. All improvements were guided by empirical "perf" measurements and maintain correctness relative to the baseline DP formulation.

1 Assumptions

The following assumptions guide the correctness, performance, and hardware compatibility of the final implementation.

1.1 Hardware Assumptions

- The CPU supports **AVX2** (256-bit SIMD).
- The CPU supports **OpenMP** for multithreading.
- A typical cache hierarchy is assumed: L1 $\approx$ 32 KB, L2 $\approx$ 256–512 KB, L3 in MB range.
- Memory alignment of at least 32 bytes is available via `posix_memalign`.
- The system allows high-bandwidth sequential memory access.

1.2 Algorithmic Assumptions

- Smith–Waterman local alignment always applies $\max(0, \cdot)$ in each DP cell.
- Blocked DP assumes each block depends only on the top, left, and top-left neighboring blocks.
- Scoring scheme:
  - 32-bit mode: MATCH = 2, MISMATCH = -1, GAP = -2.
  - 16-bit mode: uses unsigned arithmetic; mismatch and gap are penalties.
- Hybrid precision switching:

$$\text{If } N \leq 100000 \text{ use 16-bit SIMD, else use 32-bit SIMD.}$$

- Intermediate DP values never overflow 16-bit range for small and medium inputs.

1.3 Memory Model Assumptions

- DP matrix is computed in fixed-size blocks (tiles), not stored globally.
- Only block boundary rows and columns (`H_horizontal` & `H_vertical`) are stored globally.
- SIMD buffers and working arrays are aligned to 32 bytes.

2 Compiler Flags Used

The code is compiled using:

```
gcc -O3 -mavx2 -fopenmp -march=native -o sw_simd_scan_v2 gemini_ultra_2.c
```

2.1 Explanation of Flags

- **-O3**: Enables aggressive compilation optimizations, including loop unrolling and vectorization.
- **-mavx2**: Allows use of AVX2 intrinsics for 256-bit SIMD instructions.
- **-fopenmp**: Enables OpenMP-based multithreading for wavefront parallelism.
- **-march=native**: Uses all CPU-specific instruction sets available on the host machine.
- **-DBLOCK_SIZE=256**: Sets tiling block size for cache-friendly DP execution.

3 Tunable Build-Time Parameters

- **BLOCK_SIZE = 256**: Chosen to fit DP tiles into L1/L2 caches.
- **ALIGN = 32**: Required for AVX2 load/store alignment.
- **CUTOFF_N = 100000**: Determines threshold for switching to 32-bit computation.

4 Iterative Optimization Journey

This section outlines the complete evolution from a naive baseline implementation to the final hybrid high-performance architecture.

4.1 Iteration 1 — Baseline Rolling-Array SW (12.40 sec)

The initial implementation used a classic 2-row rolling array with $O(N)$ memory.

Perf Observations:

- Cycles: $\sim$ 29B
- Instructions: $\sim$ 50B
- Branch misses: 13%
- Cache misses: 18M

Performance was limited by heavy branching and a strict left-to-right dependency chain.

4.2 Iteration 2 — Scalar Loop Unrolling (5.02 sec)

Unrolling the inner loop by 8× improved ILP and reduced mispredictions.

Improvements:

- Time: 12.4s → 5.0s
- Branch misses: 13% → 0.07%
- IPC: 1.7 → 3.88

4.3 Iteration 3 — Partial AVX2 Vectorization (4.20 sec)

SIMD accelerated diagonal and upward transitions, but the left dependency forced partial scalar fallback.

4.4 Iteration 4 — Sequence Profile + Aligned SIMD (3.20 sec)

Scoring profiles removed MATCH/MISMATCH branching inside the hot loop.

Result: Instruction count reduced to 20B; memory access aligned.

4.5 Iteration 5 — Blocked Wavefront Parallelism (1.00 sec)

Switched to 256×256 blocked DP with anti-diagonal wavefront parallelism.

Why It Works:

- Blocks on the same diagonal are independent.
- Each block fits in L1/L2 cache.
- Greatly reduces global memory bandwidth.

Result: 12× speedup, 2.49 GCUPS.

4.6 Iteration 6 — Full SIMD Prefix-Scan Fix (0.818 sec)

Implemented a complete SIMD left-gap propagation using register shifting and gap offsets.

Outcome:

- Time: 1.00 → 0.818 sec
- Instructions: 18B → 12B
- IPC: ~ 1.02

4.7 Iteration 7 — Hybrid Precision Strategy

16-bit SIMD gives maximum throughput but overflows beyond score 65535; 32-bit SIMD is slower but safe.

$$\textbf{If } N \leq 100000, \text{ use 16-bit.} \qquad \textbf{If } N > 100000, \text{ use 32-bit.}$$

Performance Summary:

- N = 100k: 16-bit → 6.92 GCUPS
- N = 200k: 32-bit → 4.43 GCUPS

5 Final Summary

- Fully blocked, cache-optimized, wavefront-parallel Smith–Waterman.
- Full AVX2 16-bit/32-bit SIMD kernels.
- Automatic hybrid switching for correctness and performance.
- Achieves 3–7 GCUPS throughput depending on sequence size.