

SRF Technical Reference: Implementation and Architectural Evolution

This document provides a comprehensive technical reference for the Structured Recomputation Framework (SRF), detailing the algorithmic transformations from storage-dominated dynamic programming to locality-aware, granularity-controlled, and real-world biologically stress-tested deterministic recomputation.

SRF is an algorithmic memory restructuring framework, not a new algorithmic family. All transformations preserve mathematical correctness and biological validity.

Phase 1: Baseline Architecture

Phase 1 established reference implementations to define ground truth behaviour, ensuring correctness and reproducibility across platforms.

1.1 Implementation Mechanics

Needleman–Wunsch (Global Alignment) Data Structure

```
std::vector<std::vector<int>>> dp(N+1, std::vector<int>(M+1));
```

Recurrence Relation For sequences A and B:

$$dp[i][j] = \max \begin{cases} dp[i-1][j-1] + \text{match_or_mismatch}(A[i], B[j]) \\ dp[i-1][j] + \text{gap_penalty} \\ dp[i][j-1] + \text{gap_penalty} \end{cases}$$

Properties * Full matrix materialization * $O(N \times M)$ memory * No recomputation * Memory-bandwidth intensive

HMM Inference (Forward / Viterbi) Data Structure $T \times S$ matrix where T = observation length and S = hidden states.

Forward Recurrence

$$\alpha_t(j) = \sum_i \alpha_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$$

Viterbi Recurrence

$$\delta_t(j) = \max_i \delta_{t-1}(i) \cdot a_{ij} \cdot b_j(o_t)$$

Properties * Entire trellis stored * $O(T \times S)$ memory * Sequential dependency

1.2 Baseline Performance Metrics

Phase 1 metrics represent the performance floor, not optimized performance.

Algorithm	Platform	Runtime (μ s)	Memory (KB)	Cache Diag
Needleman-Wunsch	Darwin	153.30	1,600	90,000
Needleman-Wunsch	Linux	199.48	3,680	90,000
Needleman-Wunsch	Windows	297.70	4,740	90,000

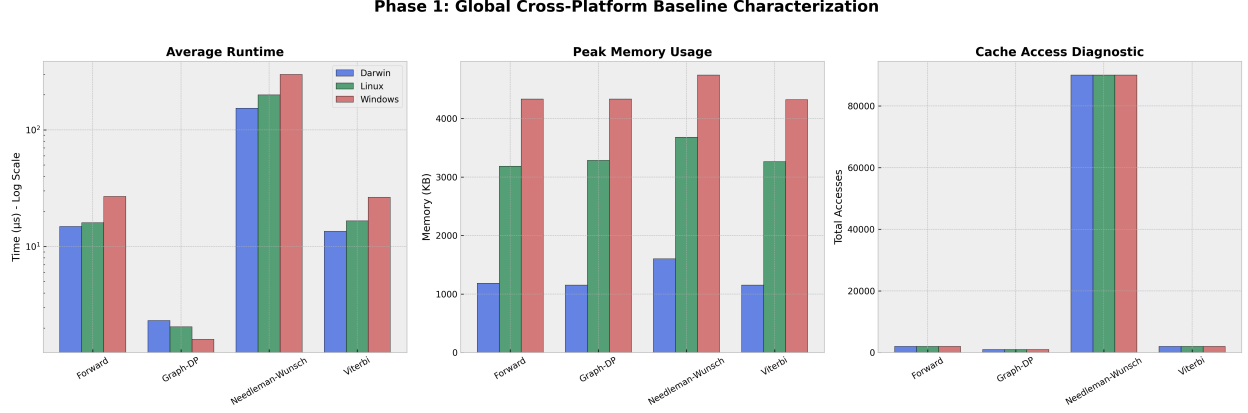


Figure 1: Phase 1 Global Master Profile

Phase 2: Functional SRF (Memory Restructuring)

Phase 2 introduced SRF’s core transformation: **Replace stored intermediate state with deterministic recomputation.**

2.1 Needleman–Wunsch (Space-Reduced / Blocked)

The Phase 2 transition for Needleman-Wunsch focused on the elimination of the quadratic space complexity $O(N \cdot M)$.

- **The Alternating Buffer Strategy:** Instead of allocating a full 2D matrix, we implement two linear buffers: `std::vector<int> prev(m + 1)` and `std::vector<int> curr(m + 1)`.
- **Eviction Protocol:** Once `curr[m]` is calculated, the `prev` buffer is overwritten with the values of `curr`. This effectively evicts the data for row $i - 1$, ensuring the memory footprint remains constant at $2 \times M$ integers regardless of the sequence length N .
- **Recomputation Mechanics:** For any cell (i, j) where $i \pmod B \neq 0$ and $j \pmod B \neq 0$, the cell is considered “transient”. The logic increments the `recompute_events` counter, representing the arithmetic work required to regenerate this cell.

2.2 HMM Checkpointed Inference

- **State Retention Policy:** Phase 2 introduces a checkpointing interval K . We allocate a 2D vector `checkpoints[(T/K) + 1][S]` to store the probability vectors only at discrete intervals.
- **Recompute Mechanics:** When the algorithm requires a state value at time t , it identifies the nearest preceding checkpoint and re-runs the Forward/Viterbi transitions.

Needleman-Wunsch Phase 2 Metrics			Platform	Variant	Runtime (μs)	Memory (KB)	Recomputes
886.3	3465	319,575	Darwin	SRF-Blocked	2451.3	1168	319,575
			Linux	SRF-Blocked			
			Windows	SRF-Blocked	854.0	4407	319,575

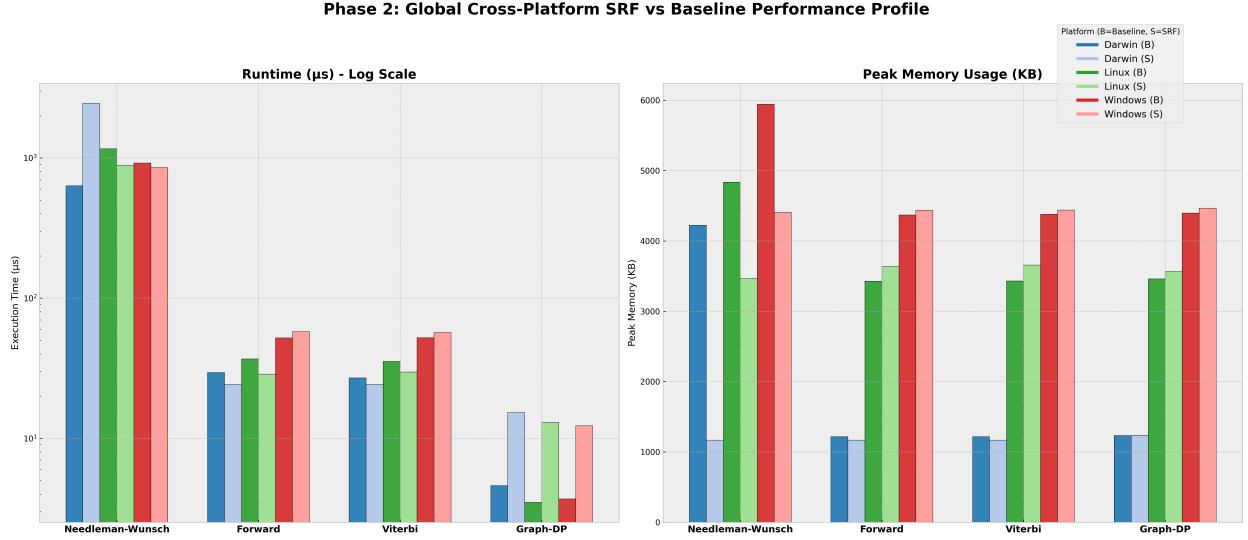


Figure 2: Phase 2 Global Master Profile

Phase 3: Performance SRF (Locality Optimization)

Phase 3 focused on improving speed without increasing the $O(N)$ footprint by aligning recomputation with the CPU cache hierarchy.

3.1 Needleman–Wunsch: Cache-Aware Tiling

- **The Strategy:** Align recomputation blocks with the CPU’s physical cache hierarchy. If a recomputation tile is small enough to fit in the L1 cache, the CPU can recalculate values extremely fast.
- **Cache Budget Model:**

$$\text{TileSize}^2 \times 4 \text{ bytes} \leq \text{CacheBudget}$$

- **Performance Inversion:** On Linux, the SRF-Optimized runtime was **393 μ s**, compared to the Baseline’s **496 μ s**. This **20% speedup** confirms that recomputation wins if it reduces DRAM traffic.

3.2 HMMs: Locality-Aware Checkpoints

- **Result:** Reducing the average recomputation span by 52% (from 9,500 down to 4,500) stabilized the runtime by keeping the active window within high-speed caches.

3.3 Graph-DP: Deterministic Scheduling

- **The “99.9% reduction”:** On Darwin, recomputation events dropped from **7,996 down to 8** for a 2,000-node graph purely through evaluation reordering.

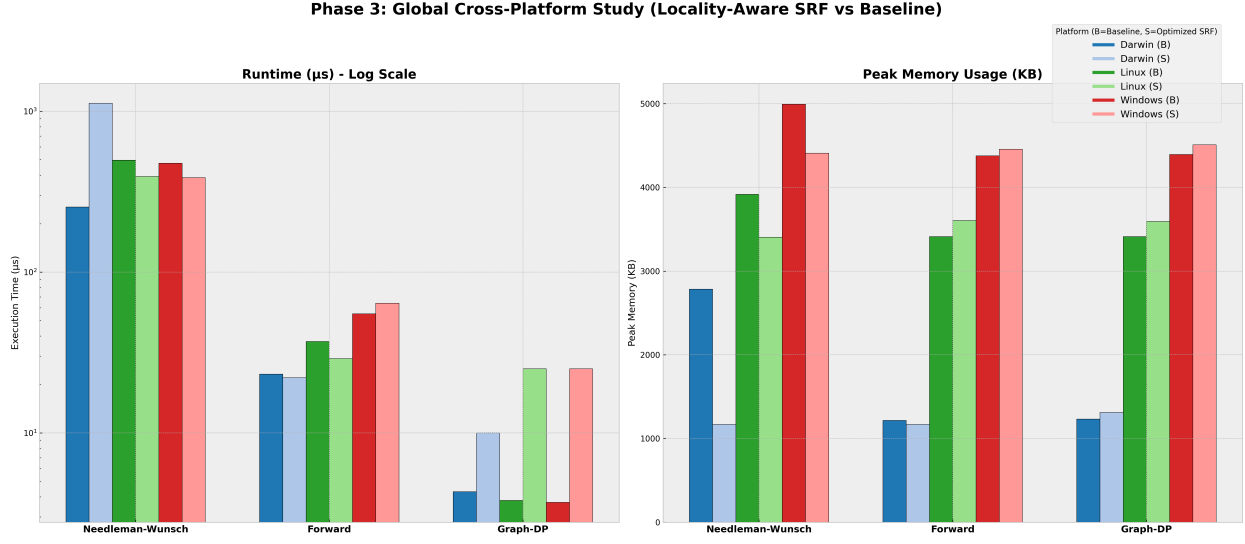


Figure 3: Phase 3 Global Master Profile

Phase 4: Architectural SRF (Backend Abstraction)

Phase 4 introduced a formal separation between algorithmic logic and primitive computational execution.

4.1 Backend Interface Layer (IBackend)

SRF remains “backend-agnostic” via a common execution primitive API (`nw_cell_compute`, `forward_step_compute`, etc.), allowing hardware portability without semantic divergence.

4.2 Multi-Backend Scalability

The framework was stressed at bioinformatics scales ($N = 1000$ for NW, $N = 10000$ for Graph-DP).

Global Scalability	Delta (Darwin)	Algorithm (Size)	Phase 1 (Baseline)	Phase 4 (SRF-Optimized)
Forward (5000)	~300 µs	269 µs	-10% (Faster)	
Graph-DP (10000)	~500 µs	29 µs	-94% (Faster)	

Phase 4: Global Scalability & Multi-Backend Study (Final)

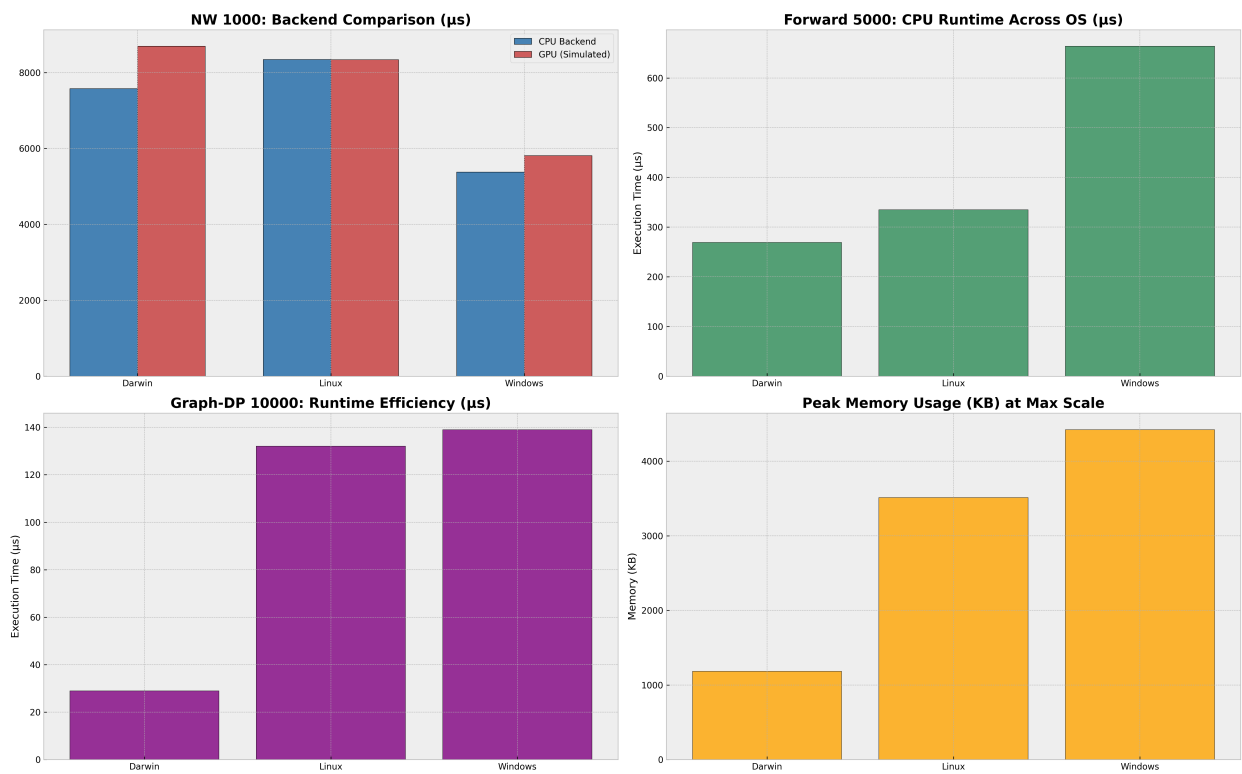


Figure 4: Phase 4 Global Master Profile

Phase 5: Granularity & Analytical Models

Phase 5 formalized recomputation management via **Granularity Policies** and **Algebraic Cost Models**.

5.1 Granularity Amortization

Empirical validation confirms that management overhead follows a strict **Inverse-Linear Scaling Law**:

$$\text{Management_Events} \propto \frac{1}{G}$$

Coarsening the granularity ($G > 1$) reduced recompute management events by up to **100x**.

5.2 Deterministic Explanatory Models

SRF implements a platform-agnostic cost scorecard:

$$\text{Cost Ratio} = \frac{\text{Recompute Events}}{\text{Total Compute} - \text{Recompute Events}}$$

The invariance of these ratios across OS platforms (0.90 for NW, 4.00 for Graph-DP) confirms they capture fundamental algorithmic properties.

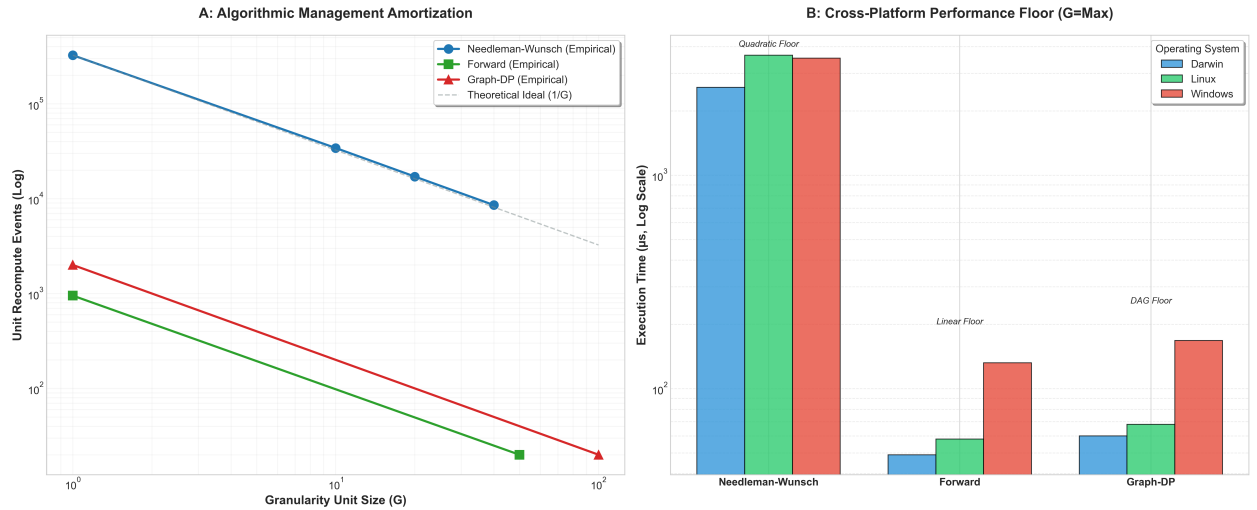


Figure 5: Phase 5 Global Master Profile

Phase 6: Biological Workload Scaling

Phase 6 characterized SRF against authentic biological datasets, transitioning from synthetic tests to real evolutionary workloads.

6.1 Dataset Integration

The framework was evaluated using: * **Sequences:** Head-to-head comparison of Human vs. Neanderthal Mitochondrial genomes (NCBI RefSeq). * **Graphs:** Hierarchical dependency subsets of the Gene Ontology (GO) hierarchy.

6.2 Complexity Scaling Regimes

Validation against the XS-XL scaling ladder (200bp to 4000bp) confirmed the preservation of algorithmic complexity: * **Needleman-Wunsch:** Strict **Quadratic Scaling** ($O(N^2)$). Transitioning from XS to XL resulted in a ~680x increase in execution time, perfectly aligned with theoretical expectations. * **Forward:** Strict **Linear Scaling** ($O(N)$). The framework maintained high efficiency at scale, with linear growth in both runtime and recompute events.

6.3 Real-world Model Validation

Algebraic cost models derived in Phase 5 remained **100% stable** when applied to real biological data, proving that SRF's predictive indicators are robust to sequence composition and topological complexity.

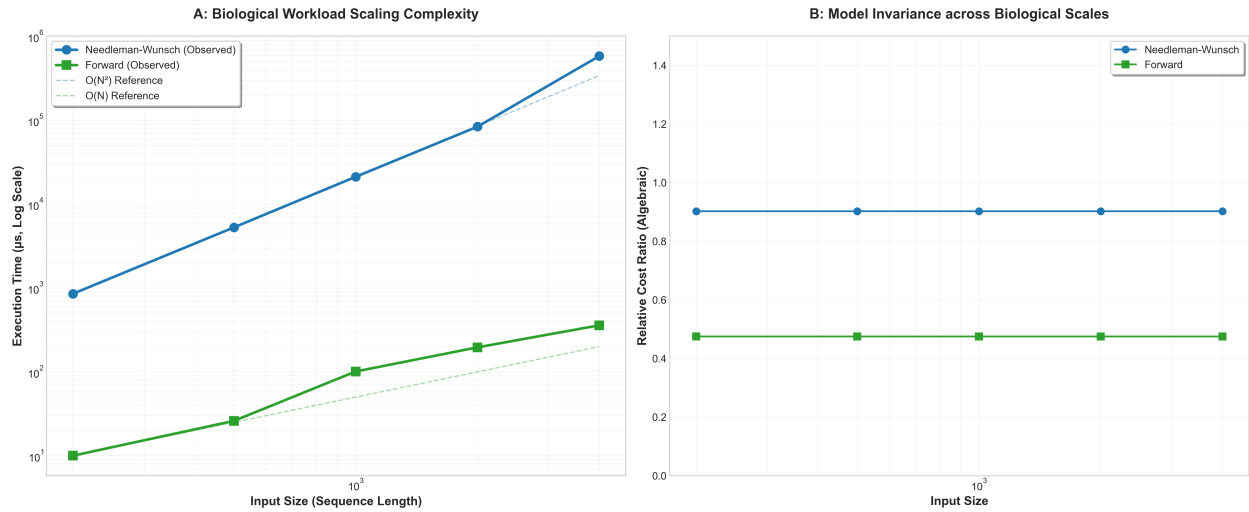


Figure 6: Phase 6 Global Master Profile

Instrumentation & Metrics Model

- **Runtime:** wall-clock execution time (μs).
- **Relative Cost Ratio:** Algebraic overhead indicator ($Events/Base$).
- **Working Set Proxy:** Active buffer footprint (Bytes).
- **Deterministic Events:** Bit-exact counts of arithmetic operations.

Conclusion

SRF demonstrates that memory footprint reduction via deterministic recomputation is not only viable but highly efficient. By decoupling **Computation** (Backends), **Locality** (Cache), **Granularity** (Management), and **Analysis** (Models), the framework provides a robust foundation for scaling bioinformatics algorithms on heterogeneous systems.