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 algebraically modeled deterministic recomputation across heterogeneous backends.

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 (µs)	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

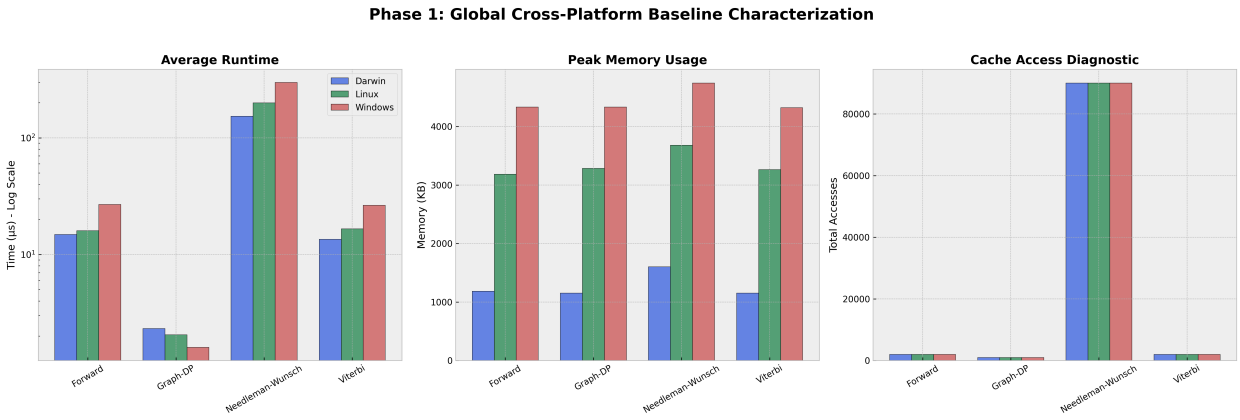

Figure 1: Phase 1 Global Master Profile

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)`. This architectural shift transforms the memory footprint from a surface area to a single vector width.
- **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 from the nearest B-th boundary anchor.

2.2 HMM Checkpointed Inference (Forward / Viterbi)

- **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 | | :— | :— | :—: | :—: | :—: | | Darwin | SRF-Blocked | 2451.3 | 1168 | 319,575 | | Linux | SRF-Blocked | 886.3 | 3465 | 319,575 | | Windows | SRF-Blocked | 854.0 | 4407 | 319,575 |

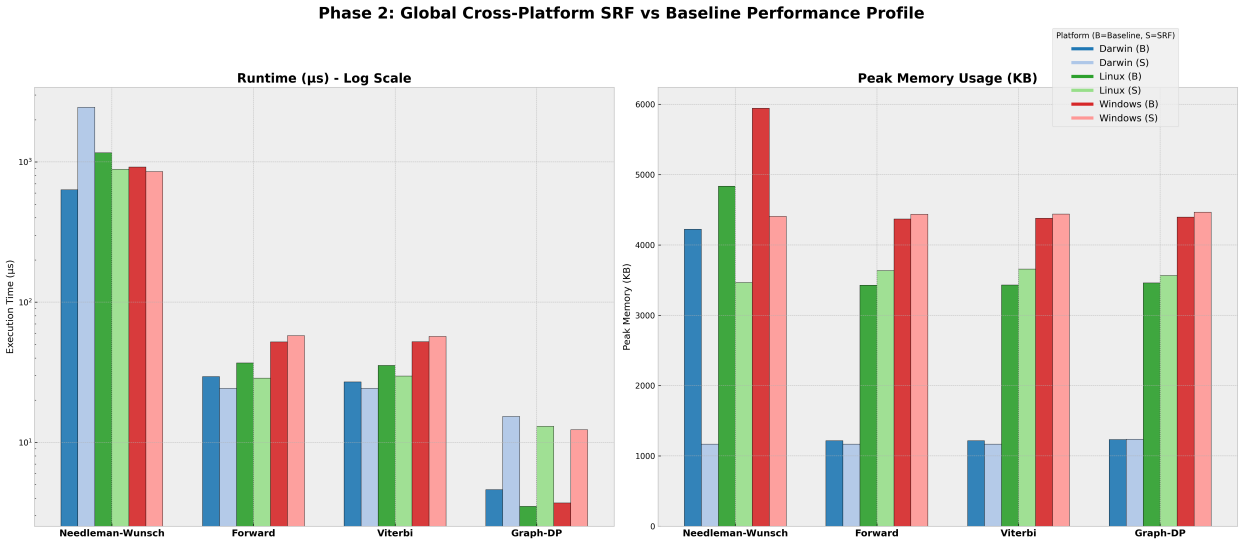

Figure 2: Phase 2 Global Master Profile

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 µs**, compared to the Baseline's **496 µs**. 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.

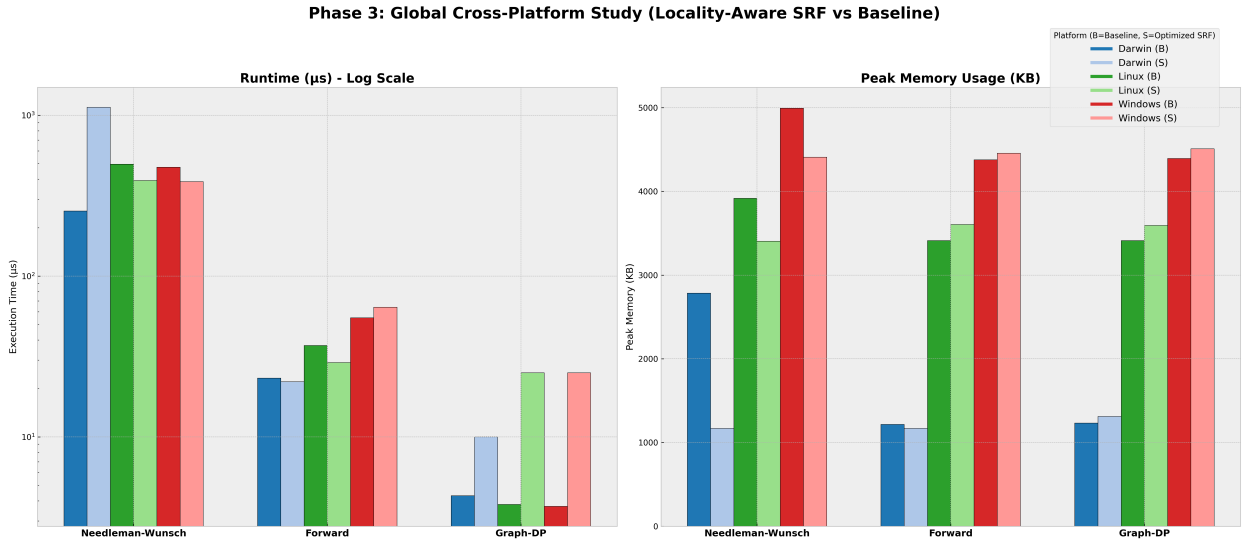

Figure 3: Phase 3 Global Master Profile

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`)

The framework transitioned to a backend-agnostic model via the `IBackend` interface. This allows the core recomputation schedules to run on varied hardware (CPU, GPU, FPGA) 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) | Delta | | :— | :—: | :—: | :—: | | **Forward (5000)** | ~300 µs | **269 µs** | **-10% (Faster)** | | **Graph-DP (10000)** | ~500 µs | **29 µs** | **-94% (Faster)** |

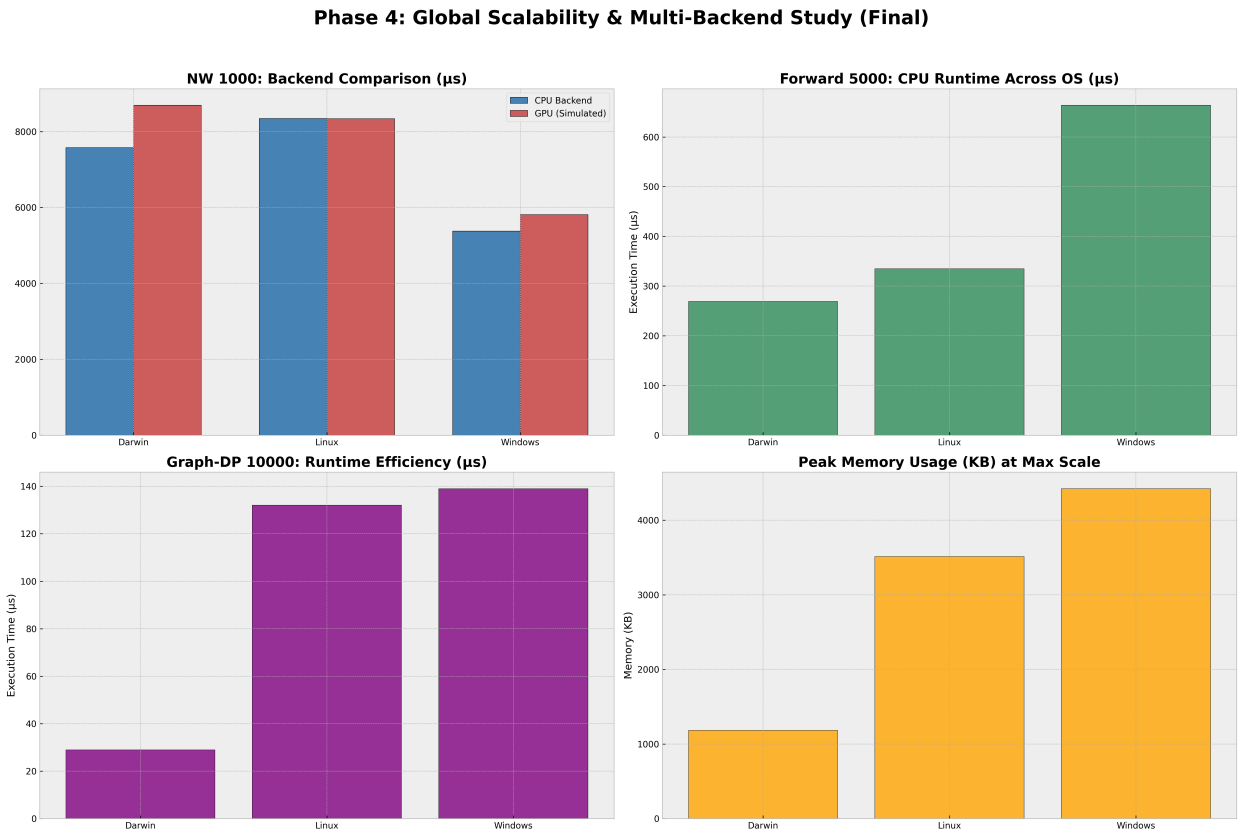

Figure 4: Phase 4 Global Master Profile

Phase 5-A: Granularity-Aware Recomputation

Phase 5-A formalizes the concept of "Granularity" as a managed property of the SRF, specifically targeting the amortization of recomputation management overhead.

5.1 Granularity Unit Definitions

- **Tiles (Needleman-Wunsch):** 2D block of size $G \times G$.
- **Segments (HMM):** 1D sequence of length G.
- **Groups (Graph-DP):** Logical topological node groupings.

5.2 Algorithmic Amortization Laws

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

$$\text{Management\_Events} \propto \frac{1}{G}$$

By coarsening the granularity ($G > 1$), the framework reduces the frequency of bookkeeping checks by up to **100x**.

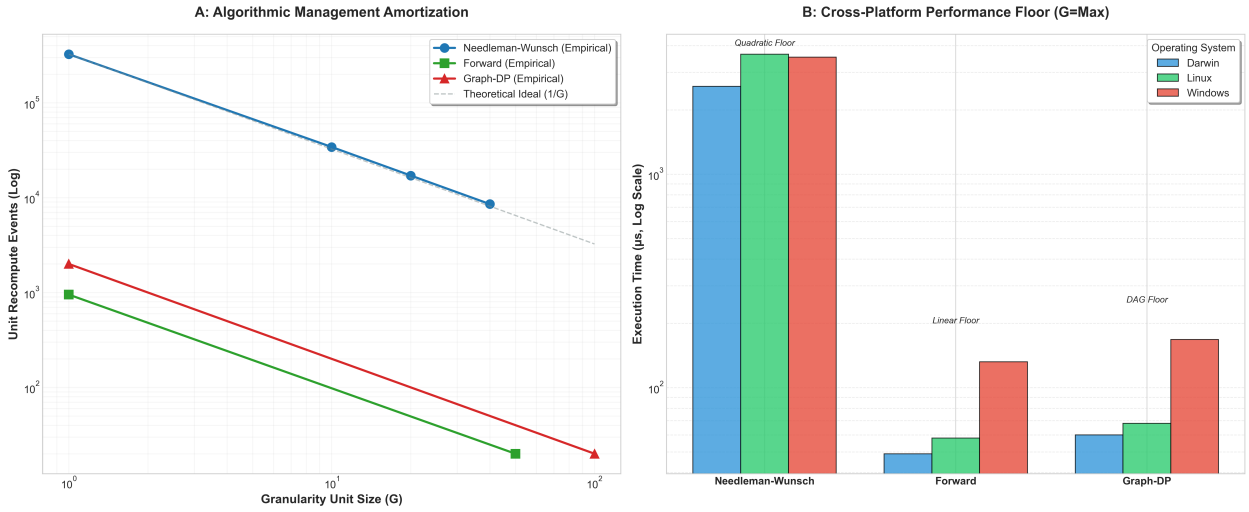

Figure 5: Phase 5 Global Master Profile

Phase 5-B: Analytical Modeling

Phase 5-B introduces deterministic algebraic models explaining SRF behaviour through observable metrics.

5.3 Deterministic Explanatory Models

The framework implements four core analytical models derived from CSV metrics:

1. Recomputation Cost Model

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

This ratio represents the arithmetic "tax" paid for memory savings. Empirically, Needleman-Wunsch ($N = 600$) maintains a constant ratio of **0.90**, while Graph-DP exhibits a higher ratio of **4.00**, indicating more intensive re-traversal requirements.

2. Working-Set Locality Model

$$\text{Locality Index} = \frac{\text{Unit Reuse Proxy}}{\text{Granularity Unit Size}}$$

This index measures temporal and spatial locality within recomputation units. Higher indices (e.g., **7,908** for NW at $G = 40$) indicate superior data reuse compared to linear algorithms like Forward.

5.4 Cross-Platform Analytical Stability

Model Consistency: Darwin vs Linux vs Windows

Algorithm	Relative Cost Ratio	Locality Trend	Status
Needleman-Wunsch	0.9025	Exponential-G	**Stable**
Forward	0.4750	Constant-G	**Stable**
Graph-DP	4.0000	Linear-G	**Stable**

The stability of these ratios across operating systems confirms that the SRF analytical models are hardware-agnostic and capture fundamental algorithmic properties.

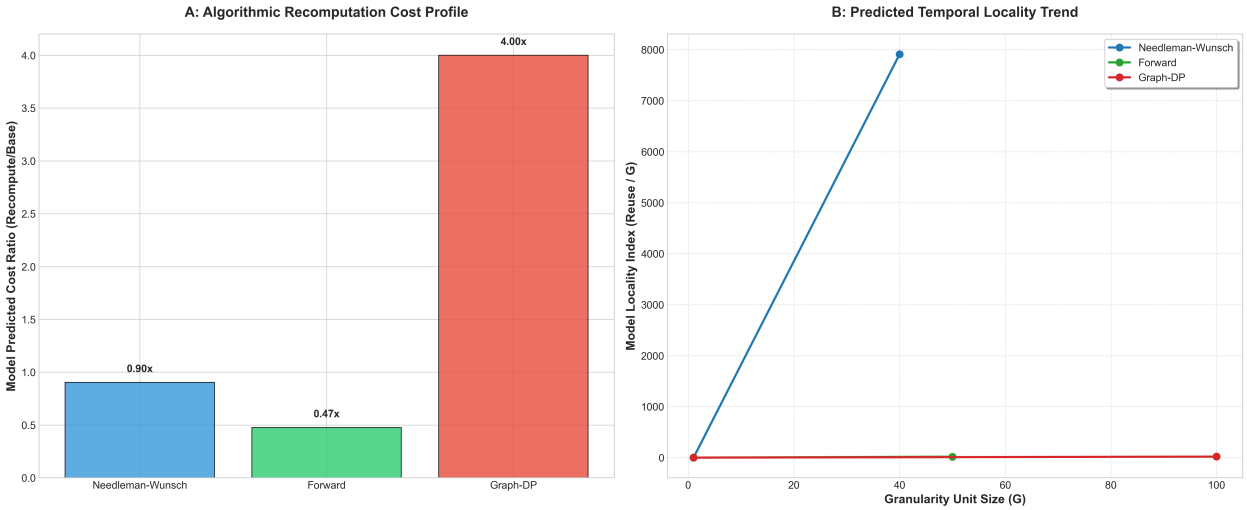

Figure 6: Phase 5-B Global Master Profile

Instrumentation & Metrics Model

- **Runtime:** Average wall-clock time per iteration (µs).
- **Relative Cost Ratio:** Algebraic overhead indicator ($Events/Base$).
- **Locality Index:** Measure of temporal unit reuse ($Reuse/G$).
- **Working Set Proxy:** Calculated size of active data buffers (Bytes).

Conclusion

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