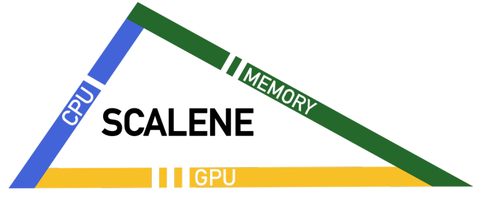
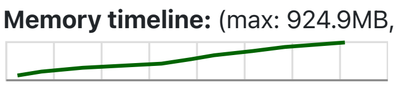


Time: Python | native | system

Memory: Python | native

Memory timeline: (max: 924.9MB, growth: 81.5%)



hover over bars to see breakdowns; click on COLUMN HEADERS to sort.

`mprof.py`: % of time = 100.0% out of 1603.1s.

TIME	MEMORY *average*	MEMORY *peak*	MEMORY *timeline*	MEMORY *activity*	COPY *(MB/s)*	GPU *util.*	GPU *memory*	LINE PROFILE *(click to reset order)* `mprof.py`
								28 `import numpy as np`
								29 `import portion`
								31 `import tskit`
								34 `def overlapping_segments(segments):`
			possible leak					39 `    S = sorted(segments, key=lambda x: x.left)`
								40 `    n = len(S)`
								48 `        left = right`
								49 `        X = [x for x in X if x.right > left]`
								50 `        if len(X) == 0:`
								51 `            left = S[j].left`
								52 `        while j < n and S[j].left == left:`
								53 `            X.append(S[j])`
								54 `            j += 1`
								56 `        right = min(x.right for x in X)`
								57 `        right = min(right, S[j + 1].left)`
								58 `        yield left, right, X`
								63 `        X = [x for x in X if x.right > left]`
			possible leak					78 `    def __init__(self, left=None, right=None, node=None, next_segment=None):`
								79 `        self.left = left`
								80 `        self.right = right`
								81 `        self.node = node`
								82 `        self.next = next_segment`
								118 `            all_edges = list(self.ts.edges())`
								119 `            edges = all_edges[:1]`
								120 `            for e in all_edges[1:]:`
								122 `                self.process_parent_edges(edges)`
								124 `                edges.append(e)`
								132 `        assert len({e.parent for e in edges}) == 1`
								135 `        for edge in edges:`
								136 `            x = self.A_head[edge.child]`
								137 `            while x is not None:`
								138 `                if x.right > edge.left and edge.right > x.left:`
								139 `                    y = Segment(`
								140 `                        max(x.left, edge.left), min(x.right, edge.right), x.node`
			possible leak					142 `                    S.append(y)`
								143 `                x = x.next`
								145 `        self.check_state()`
								152 `        is_sample = input_id in self.samples`
								162 `        for left, right, X in overlapping_segments(S):`
								174 `                for x in X:`
								175 `                    ancestry_node = x.node`
								177 `                    self.add_ancestry(left, right, ancestry_node, input_id)`
								184 `            self.flush_edges()`
								199 `    def add_ancestry(self, left, right, node, current_node):`
								200 `        tail = self.A_tail[current_node]`
								201 `        if tail is None:`
								203 `            self.A_head[current_node] = x`
								206 `            if tail.right == left and tail.node == node:`
								209 `                x = Segment(left, right, node)`
								210 `                tail.next = x`
								211 `                self.A_tail[current_node] = x`
								219 `        for child in sorted(self.edge_buffer.keys()):`
								229 `        for j in range(num_nodes):`
								230 `            head = self.A_head[j]`
								231 `            tail = self.A_tail[j]`
								232 `            if head is None:`
								233 `                assert tail is None`
								236 `                while x.next is not None:`
								237 `                    x = x.next`
								238 `                assert x == tail`
								239 `                x = head.next`
								240 `                while x is not None:`
								241 `                    assert x.left < x.right`
								242 `                    if x.next is not None:`
								243 `                        if self.ancestors is None:`
								246 `                        if x.right == x.next.left:`
								247 `                            assert x.node != x.next.node`
								248 `                    x = x.next`
								270 `    ts = tskit.load(sys.argv[1])`
								273 `    samples = ts.samples()`
								278 `    ancestors = [u.id for u in ts.nodes() if u.time == census_time]`

TIME	MEMORY *average*	MEMORY *peak*	MEMORY *timeline*	MEMORY *activity*	COPY *(MB/s)*	GPU *util.*	GPU *memory*	FUNCTION PROFILE *(click to reset order)* `mprof.py`
								34 `AncestorMap.overlapping_segments`
								78 `Segment.__init__`
								116 `AncestorMap.link_ancestors`
								128 `AncestorMap.process_parent_edges`
								147 `AncestorMap.merge_labeled_ancestors`
								199 `AncestorMap.add_ancestry`
								213 `AncestorMap.flush_edges`
								226 `AncestorMap.check_state`