Whippet: A practical memory management upgrade for Guile & beyond

2 February 2025 – FOSDEM ’25

Andy Wingo

Igalia, S.L.

Agenda

- The big idea
- The results
- The future

The big idea

Whippet is a practical memory management upgrade for Guile & beyond

A practical **memory management** upgrade for Guile & beyond

```
struct gc_options *options = NULL;
struct gc_stack_addr *stack_base = NULL;
struct gc_heap *heap;
struct gc_mutator *mut;
void *event_listener_data = NULL;

gc_init(options, stack_addr, &heap, &mut,
        GC_NULL_EVENT_LISTENER, event_listener_data);

void *obj = gc_allocate(mut, 42);
```

```
options = gc_allocate_options();
gc_options_parse_and_set_many(options,
                              getenv("GC_OPTIONS"));

struct gc_mutator_roots roots; // Embedder-defined
gc_mutator_set_roots(mut, &roots);

// For generational configurations
gc_write_barrier(mut, obj, obj_size, edge, new_val);

// For cooperative safepoints
gc_safepoint(mut);

// For collectors that don't require copying
gc_pin_object(mut, ref);
```

```
static inline void
gc_trace_object(struct gc_ref ref,
                void (*visit)(struct gc_edge edge,
                              struct gc_heap *heap,
                              void *visit_data),
                struct gc_heap *heap,
                void *trace_data,
                size_t *size) { /* ... */ }

static inline void
gc_trace_mutator_roots(struct gc_mutator_roots *roots,
                       void (*trace_edge)(struct gc_edge edge,
                                          struct gc_heap *heap
                                          void *trace_data),
                       struct gc_heap *heap,
                       void *trace_data) { /* ... */ }
```

A practical memory management **upgrade** for Guile & beyond

Whippet: An upgrade relative to BDW-GC

Performance: Bump-pointer allocation, better parallelism

Features: Ephemerons and finalizers that work

Behavior: Choice of workload-appropriate collectors

Memory use: Compaction, adaptive heap sizing (membalancer)

Whippet: An upgrade with a migration path

Bridge, *n.*: Construction with two ends and a path in between

Whippet: a GC library with compile-time abstraction over embedder needs and collector construction

Collector variants: MMC, PCC, BDW

MMC collector has optional conservative tracing

- stack roots
- global roots
- (optionally) intra-heap edges

MMC

Mostly-marking collector

MMC = nofl space + lospace

Nofl space

“No free-list”

For objects less than 8192 bytes

Bump-pointer allocation

Excellent parallelism

Mostly-marking (Immix-derived), occasionally compacting

12% overhead

Pinning (transitively due to conservative roots, or permanently)

Optionally generational (sticky-mark)

Lospace

“Large object space”

`mmap` allocation, freelist, deferred release

Optionally generational (sticky-mark)

Whippet: An upgrade relative to bespoke GCs

Language run-times often get stuck with their GC

Whippet’s compile-time API abstraction enables evolution

PCC

Parallel copying collector

PCC = copy space + lospace

Copy space

For objects less than 8192 bytes

Bump-pointer allocation

Excellent parallelism

Always compacting

100% overhead

Generational PCC

Generational PCC = copy space + copy space + lospace

2 MB nursery per processor active in last cycle

1 survivor cycle for copy space, 0 for lospace

Field-logging write barrier

Nursery memory range aligned, can be quick XOR check

Still has 100% overhead

BDW

Boehm-Demers-Weiser collector

Shim behind Whippet API

Different safepoint behavior: not cooperative

No support for `gc_trace_object`

Not great parallelism

Higher memory overhead than MMC

A practical memory management upgrade for Guile & beyond

Embed-only

No dependencies

C11

Hackable

Practical testbench: Whiffle

Scheme-to-C compiler: `https://github.com/wingo/whiffle`

- Ensure Whippet offers appropriate API for embedders
- Allow more test cases to be written before moving to Guile
- Handles vs stack maps

Main motivation was testing; shook out many bugs

A practical memory management upgrade for Guile & beyond

The pivot:

- Whippet API, but BDW collector
- MMC collector, with conservative roots
- Generational MMC collector (write barriers)
- Evacuating nursery?

Shout-out to NLnet foundation for helping us with this work!

A practical memory management upgrade for Guile & beyond

WebAssembly+GC-to-C: Enable standalone Guile compilation via Hoot ?

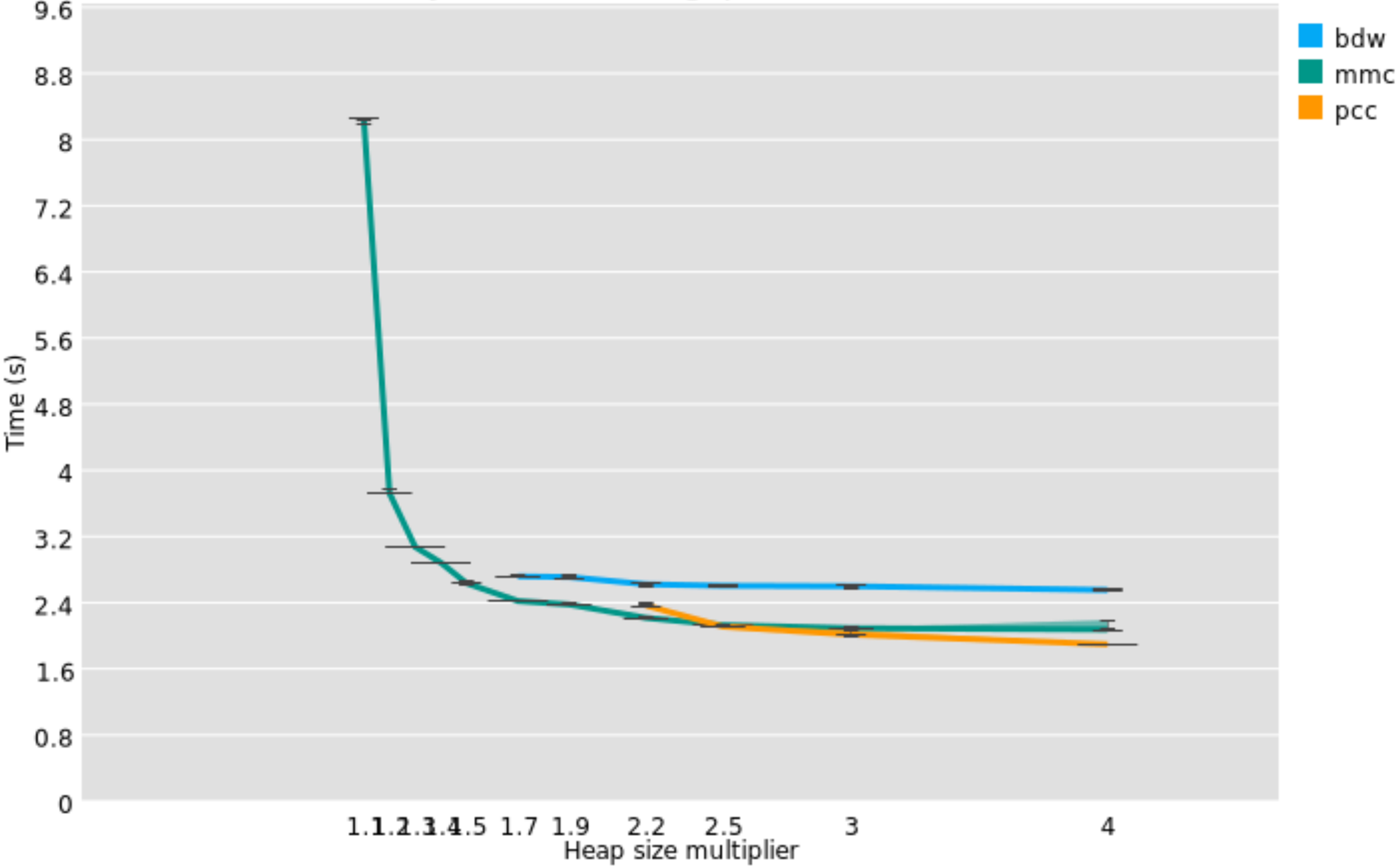
Ocaml, R, etc...

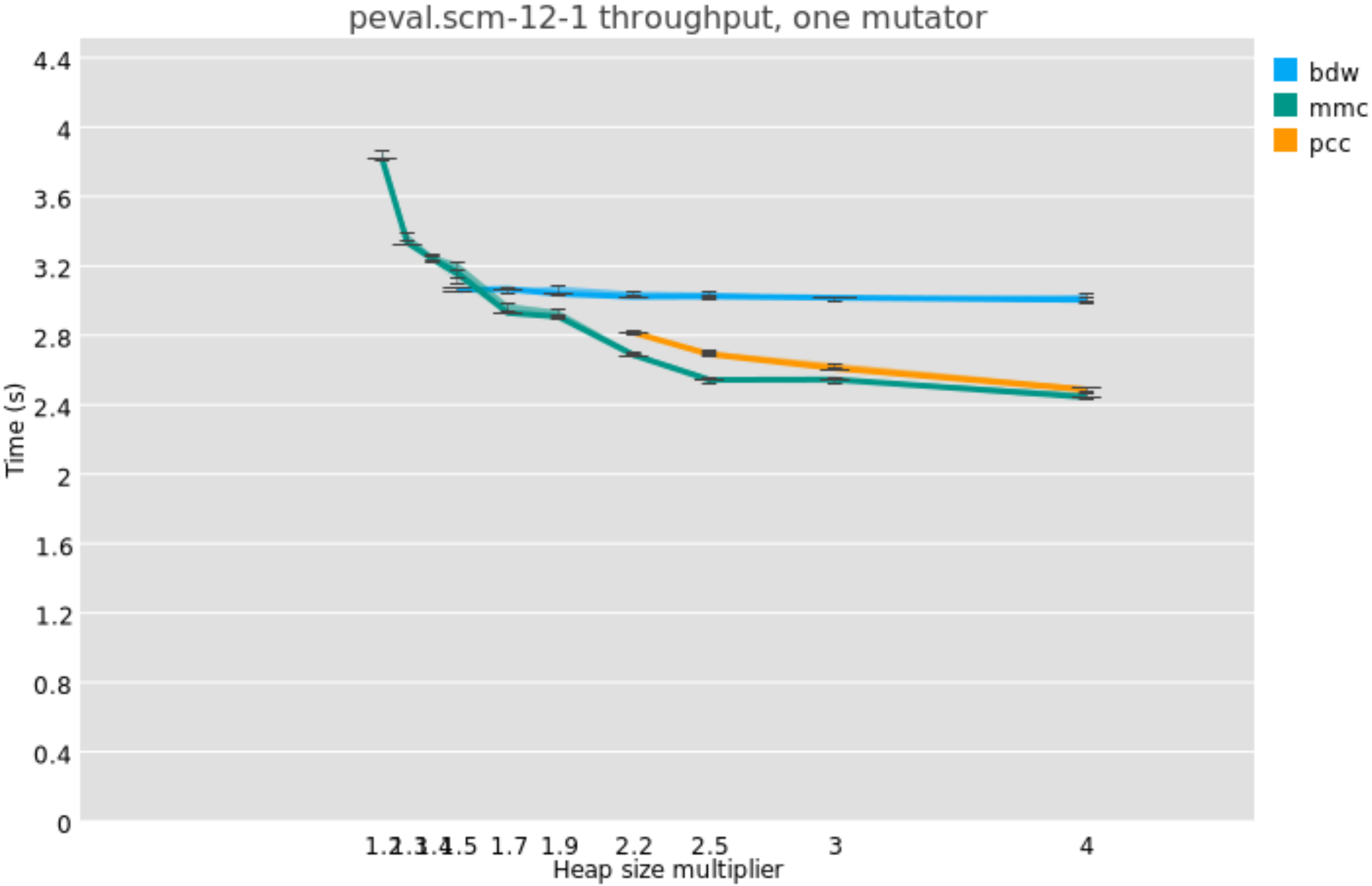
Results: What do we win with Whippet?

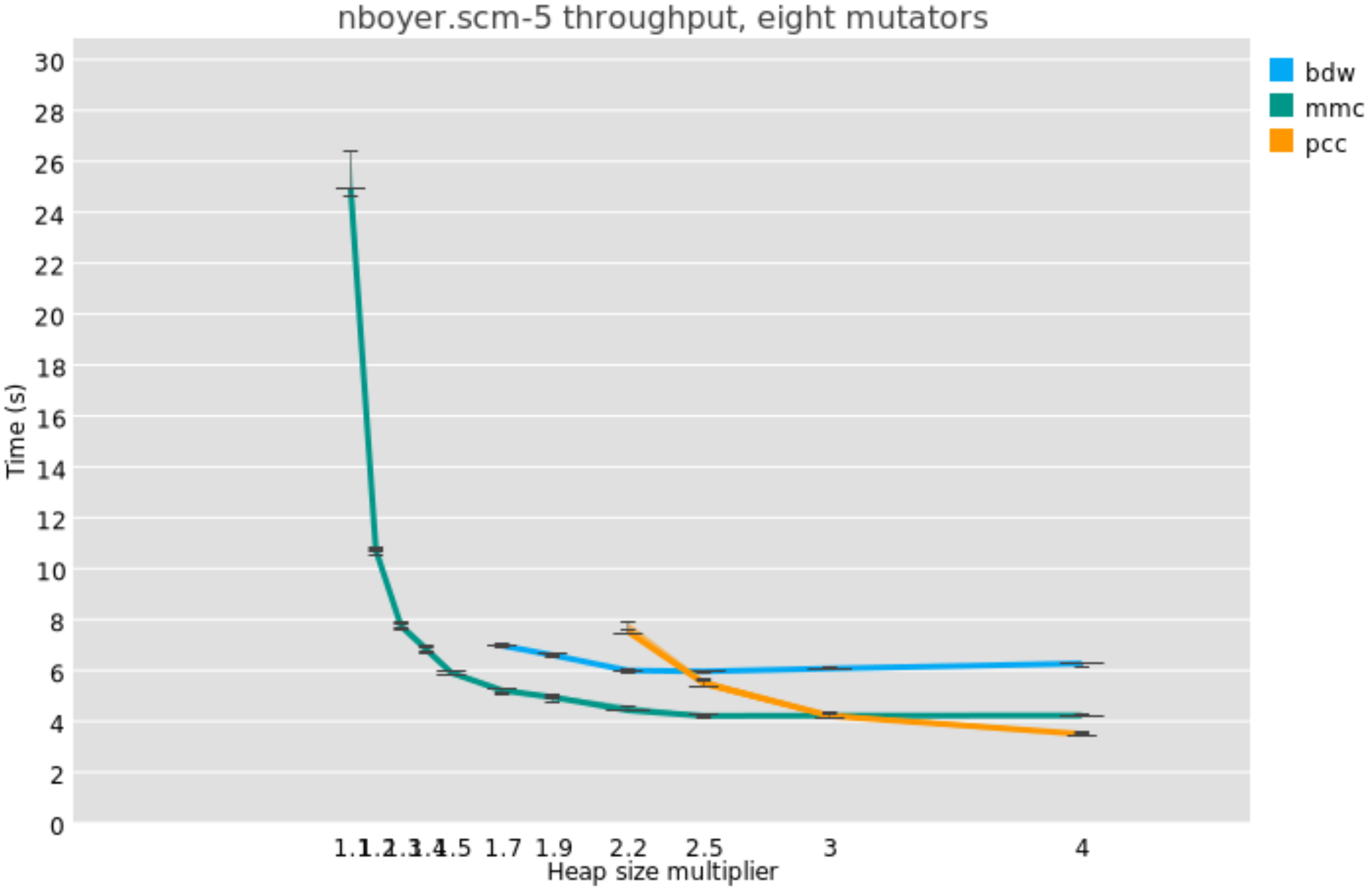
Strict throughput improvements: 20-40%

Access to smaller heap sizes: 30-50%

nboyer.scm-5 throughput, one mutator

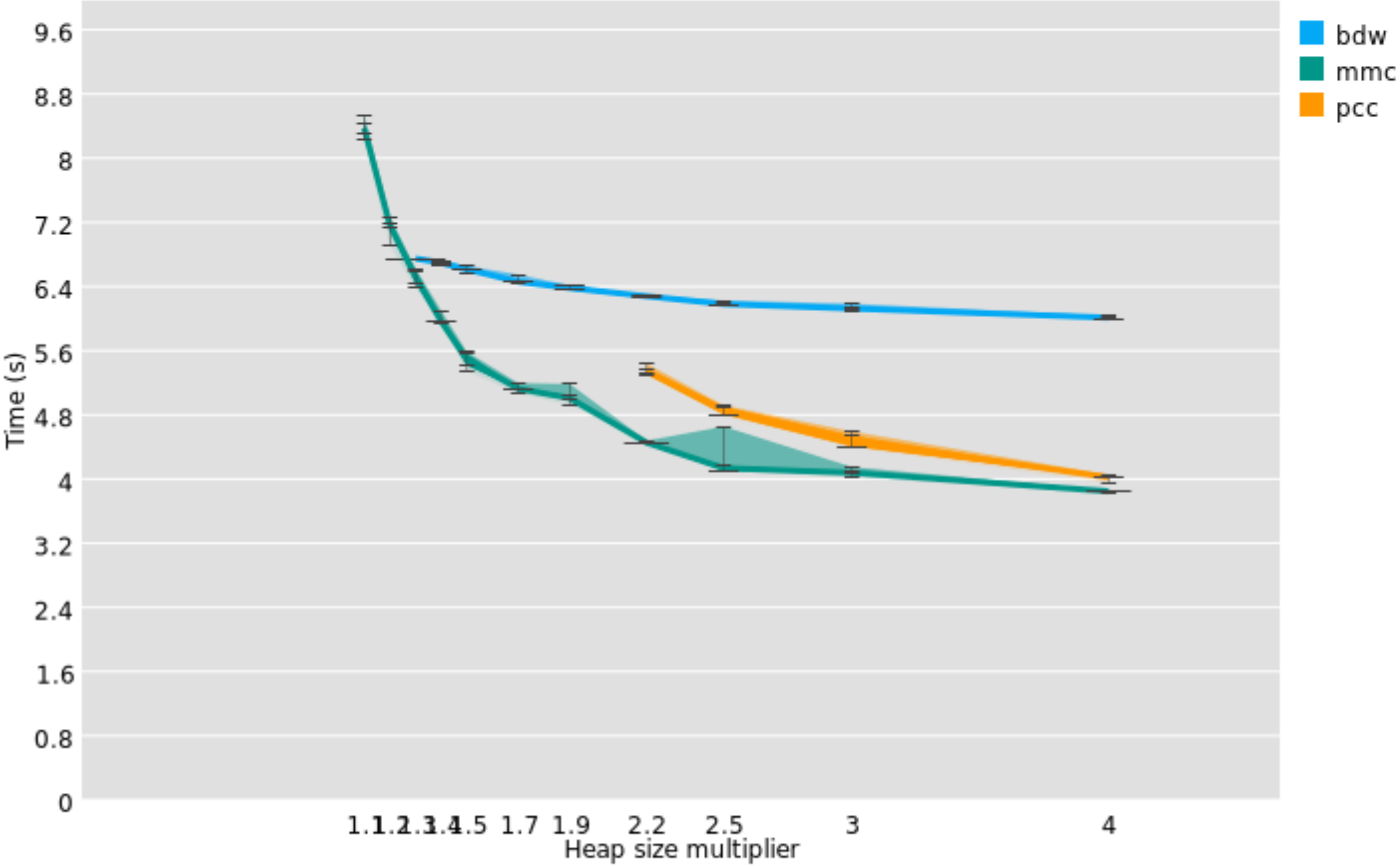



peval.scm-12-1 throughput, one mutator
Time (s)
4.4
4
3.6
3.2
2.8
2.4
2
1.6
1.2
0.8
0.4
0
1.2
1.3
1.4
1.5
1.7
1.9
2.2
2.5
3
4
Heap size multiplier
bdw
mmc
pcc



nboyer.scm-5 throughput, eight mutators

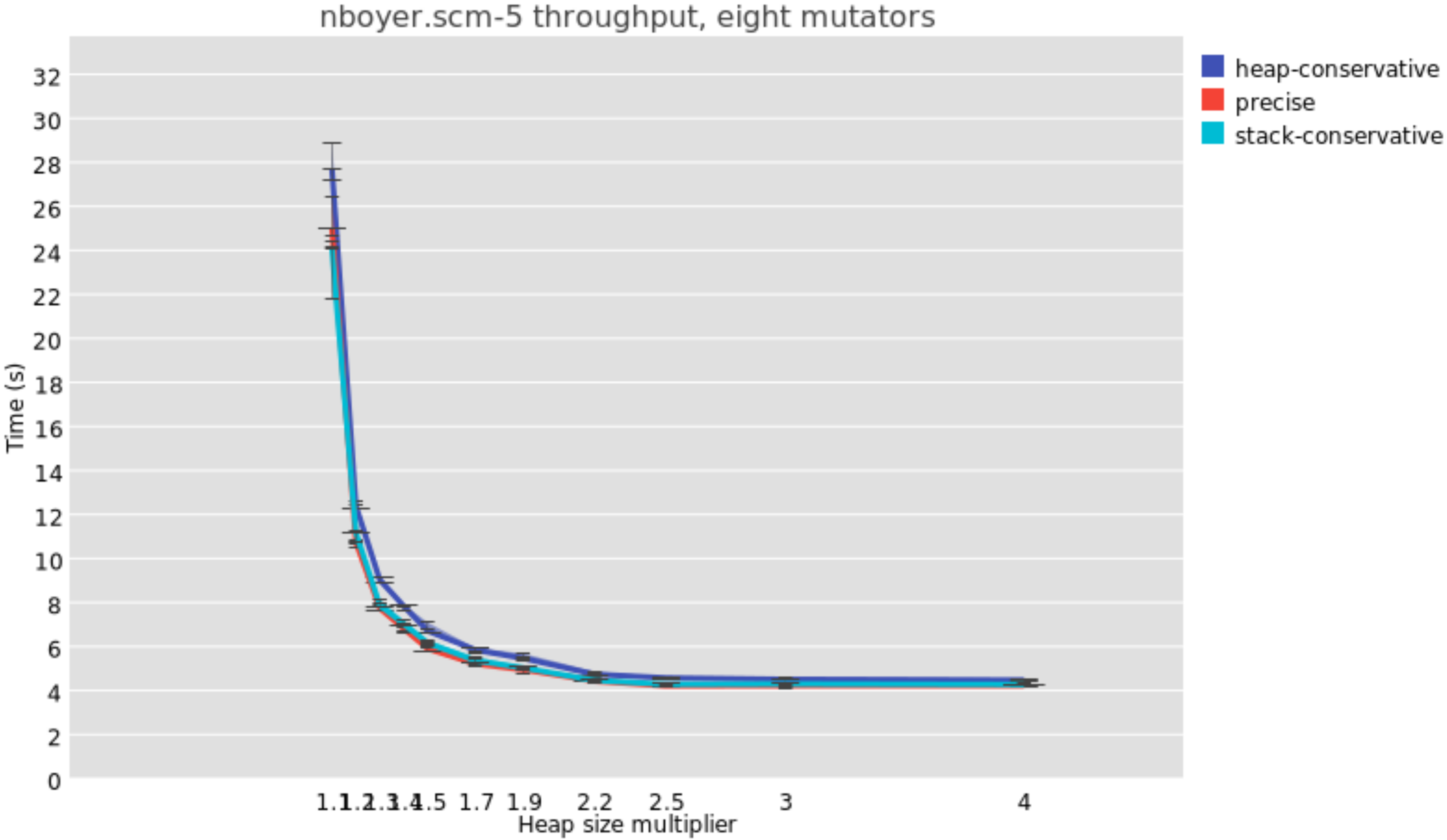
peval.scm-12-1 throughput, eight mutators

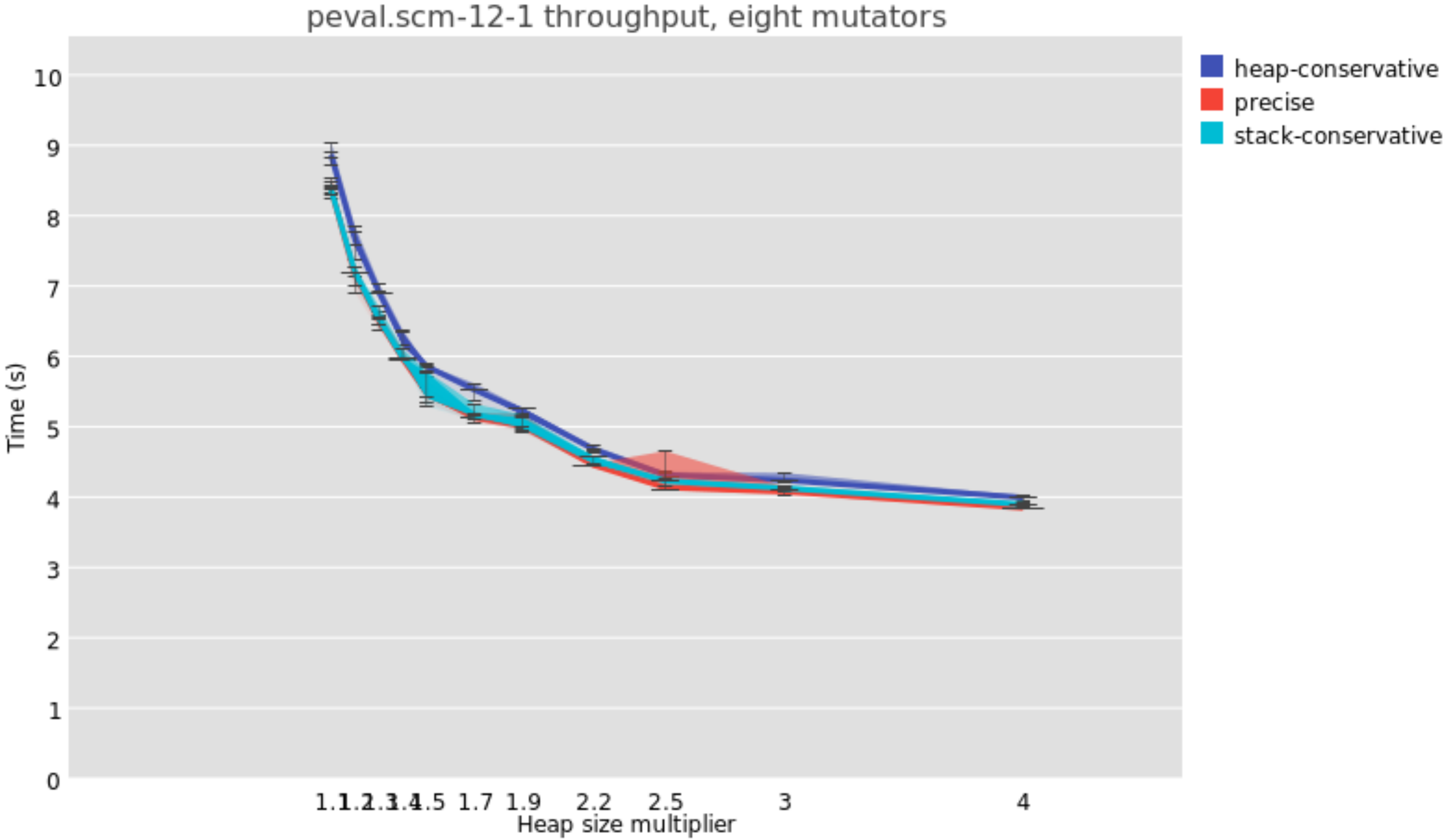


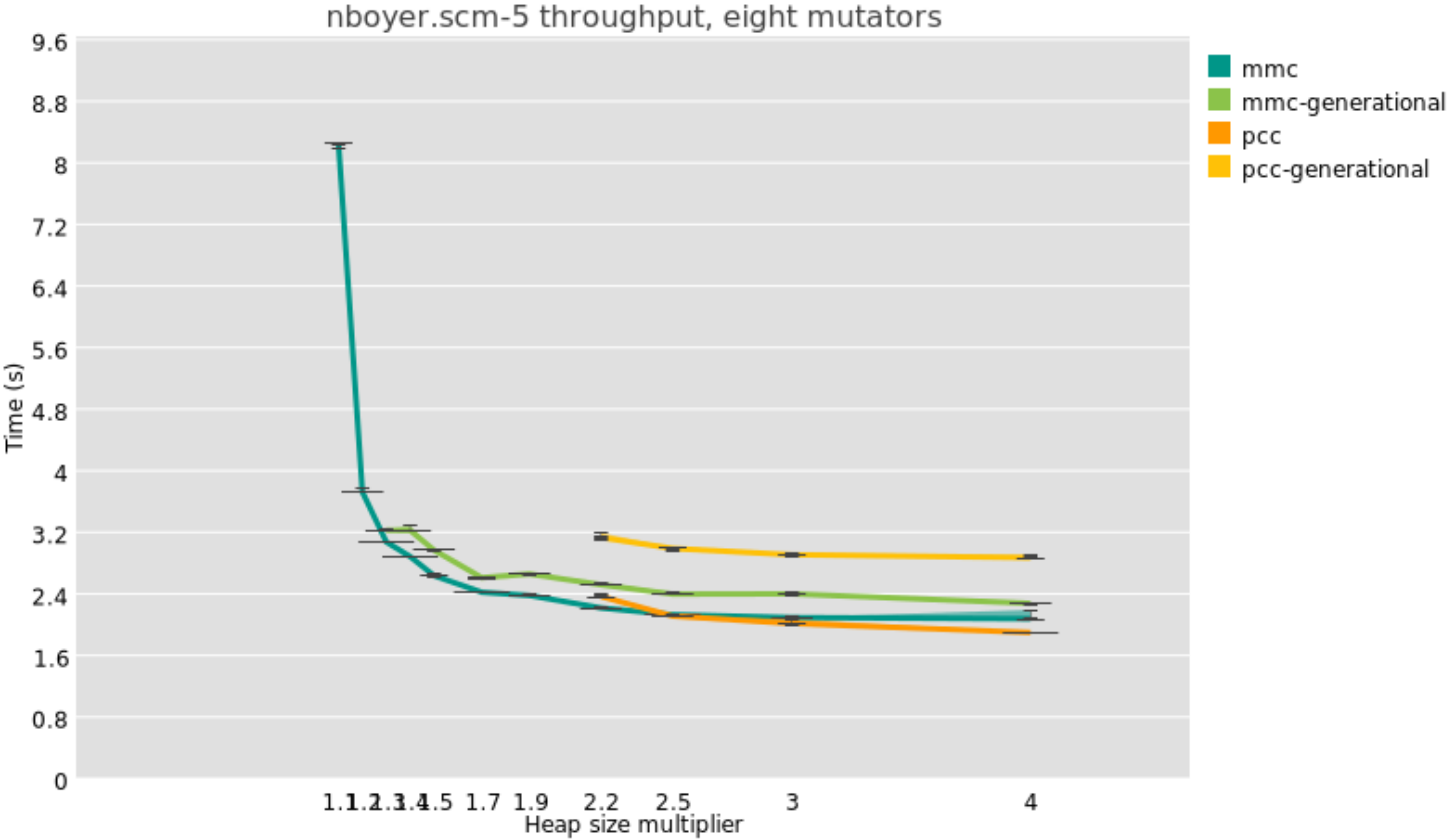
Results: What do we learn with Whippet?

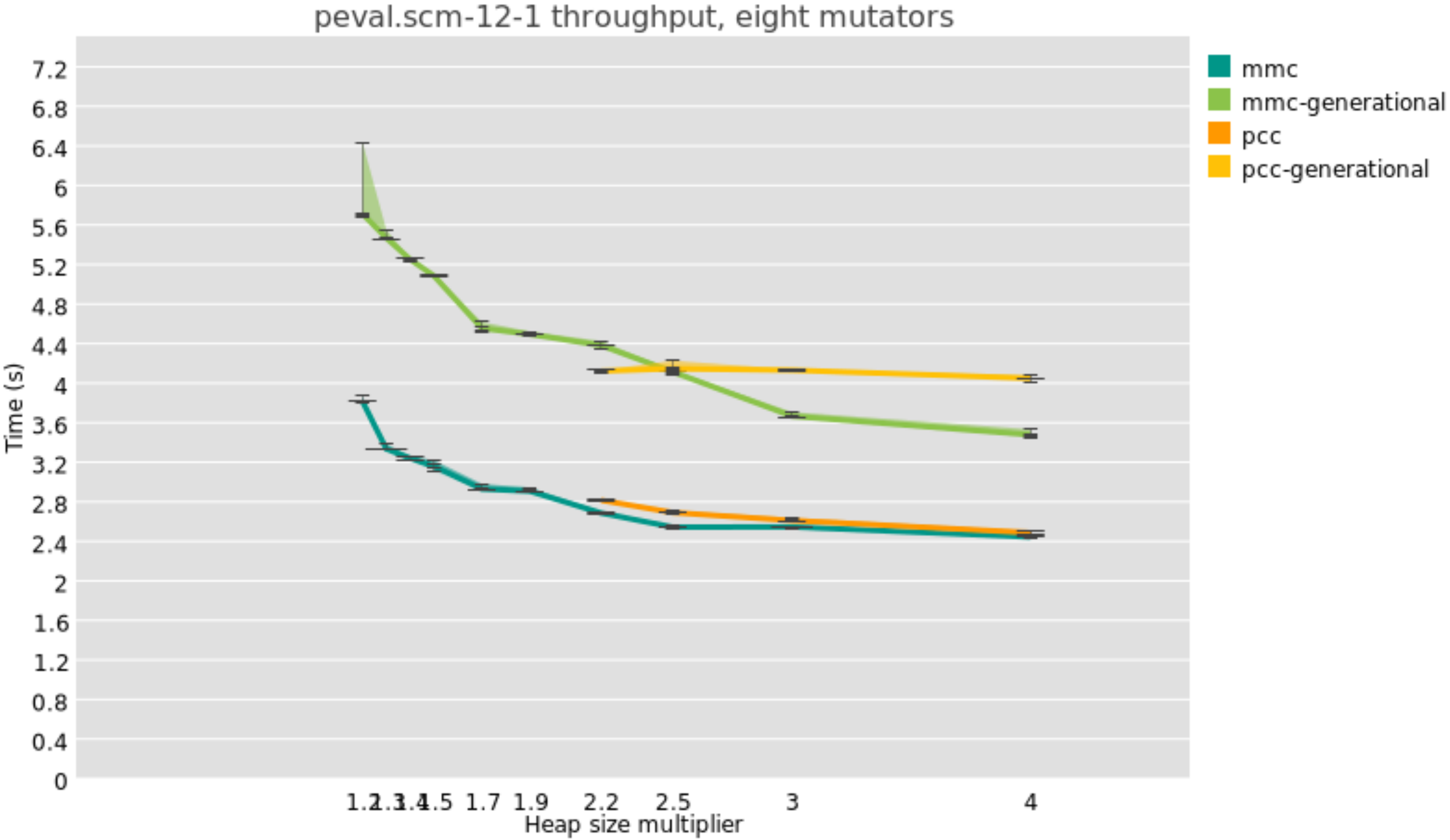
Conservative root-finding is OK

Generational GC is complicated, more tuning needed


nboyer.scm-5 throughput, eight mutators
Time (s)
0
2
4
6
8
10
12
14
16
18
20
22
24
26
28
30
32
1.1
1.2
1.3
1.4
1.5
1.7
1.9
2.2
2.5
3
4
Heap size multiplier
heap-conservative
precise
stack-conservative


peval.scm-12-1 throughput, eight mutators
Time (s)
0
1
2
3
4
5
6
7
8
9
10
Heap size multiplier
1.1
1.2
1.3
1.4
1.5
1.7
1.9
2.2
2.5
3
4
heap-conservative
precise
stack-conservative


nboyer.scm-5 throughput, eight mutators
Time (s)
9.6
8.8
8
7.2
6.4
5.6
4.8
4
3.2
2.4
1.6
0.8
0
1.1
1.2
1.3
1.4
1.5
1.7
1.9
2.2
2.5
3
4
Heap size multiplier
mmc
mmc-generational
pcc
pcc-generational


peval.scm-12-1 throughput, eight mutators
mmc
mmc-generational
pcc
pcc-generational
Time (s)
7.2
6.8
6.4
6
5.6
5.2
4.8
4.4
4
3.6
3.2
2.8
2.4
2
1.6
1.2
0.8
0.4
0
1.2
1.3
1.4
1.5
1.7
1.9
2.2
2.5
3
4
Heap size multiplier

Future

Guile, finally. This month!!!

Your language run-time?

Concurrent marking

LXR-inspired reference counting of old generation?

Try it out!

https://github.com/wingo/whippet

https://github.com/wingo/whiffle

https://nlnet.nl/project/Whippet/

wingo@igalia.com

Thanks!

Attic

New since 2023

PCC, generational PCC, precise field-logging write barriers instead of card marking, better parallelism, bug fixes, embeddability, finalizers, dynamic heap sizing (membalancer), less VMM traffic, whiffle, tests, nlnet, platform abstraction, options interface, extern space, stats, HDR histogram, renamings, nofl more eager