POWER sort

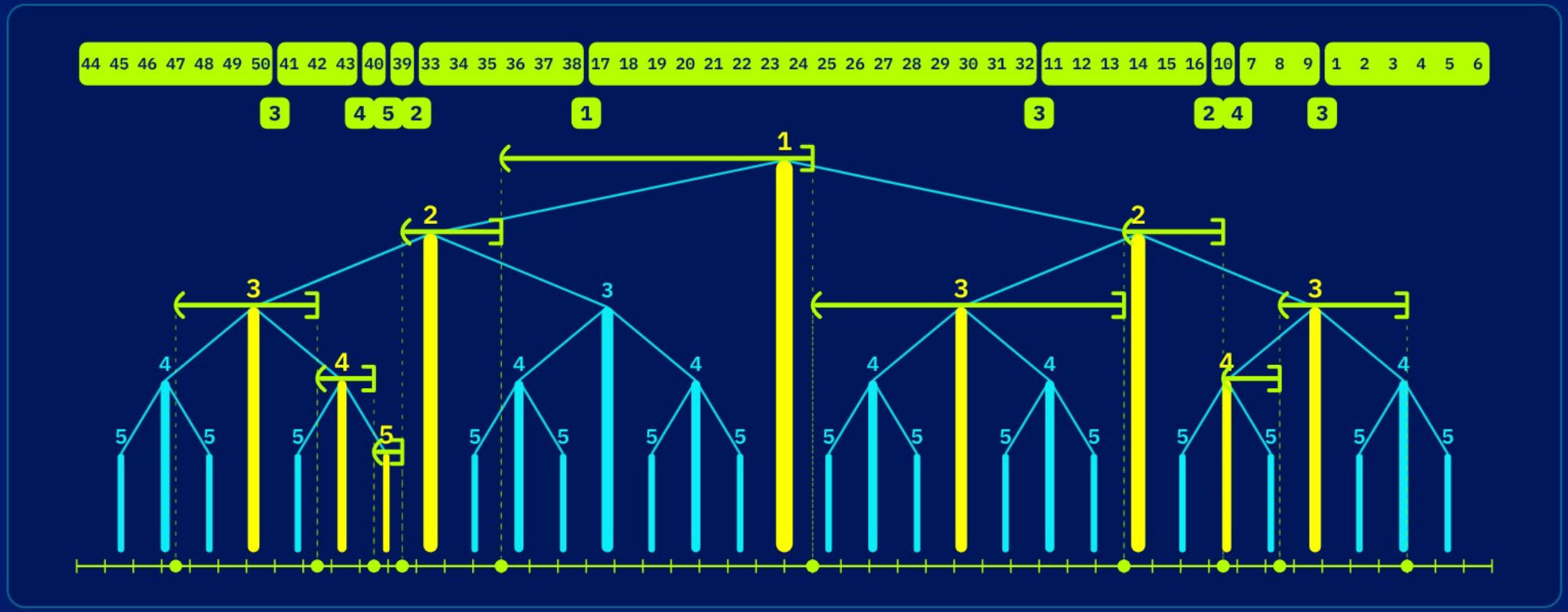
Virtual-Memory Powersort

Stable, run-adaptive mergesort with **O(√(n log n))** extra space instead of a buffer for **n/2** elements.

Finn Moltmann Tamio-Vesa Nakajima Sebastian Wild
Philipps-Universität Marburg

ESA 2026

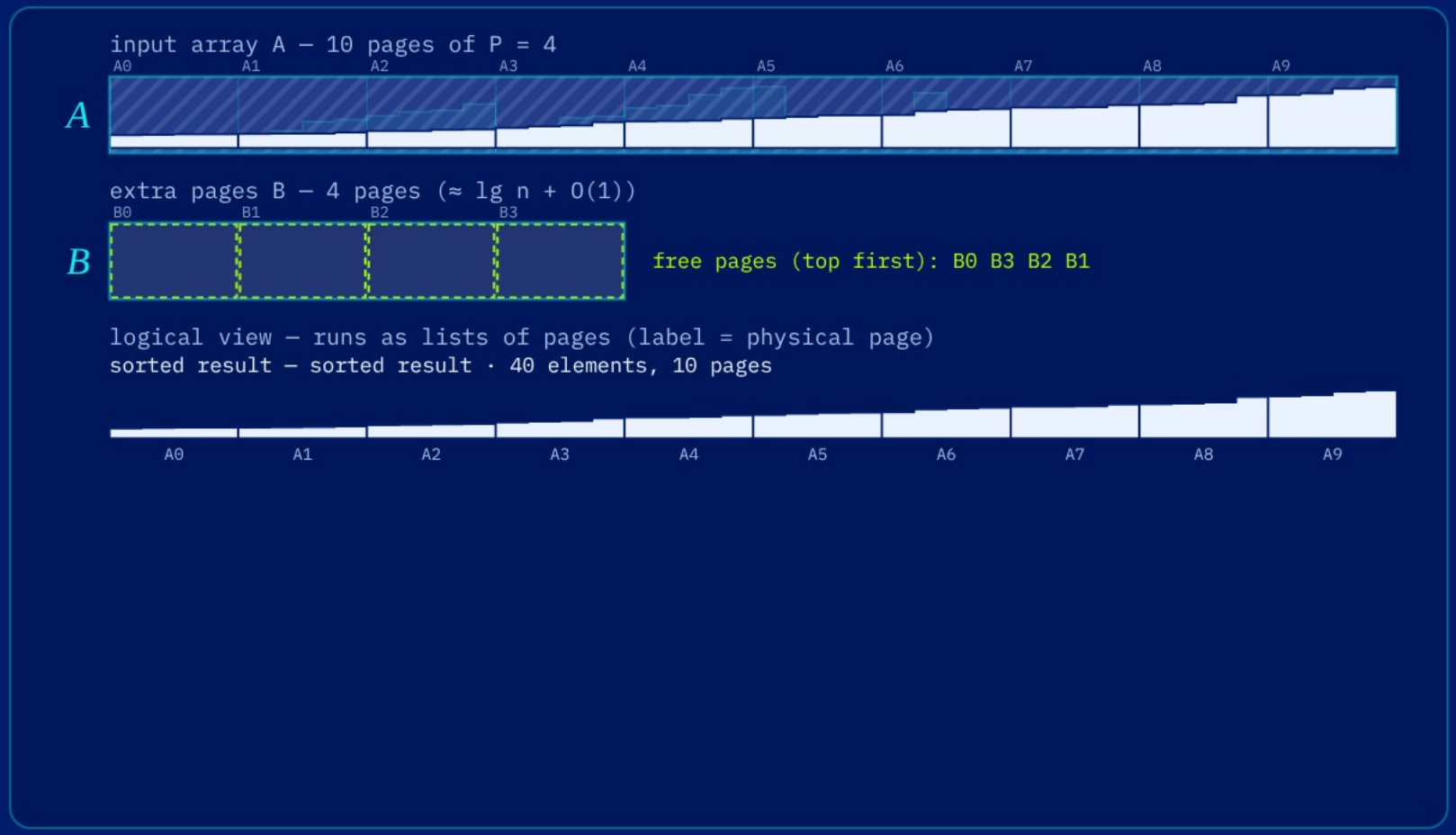
Powersort merges runs in nearly optimal order



THE CATCH

Every Timsort / Powersort implementation merges through a buffer of **n/2** elements. Merging in place exists — but is several times slower.

Runs are lists of pages, scattered over memory



▶ play ↺ restart 55 / 55

one color per run · free page · not yet scanned · n = 40, P = 4

1 2 3 4 5 6

6 · Done

A is sorted, all extra pages are free again. M + 3n moves, M + n comparisons.

> done: A is sorted, all extra pages are free again

extra space **lg n + O(1)** pages
= **O(√(n lg n))** elements
moves **M + 3n** · comparisons **M + n**

Quadratically less memory, negligible slowdown

extra memory · int (36 MB)

CPython-style — 18 MB · 50 %

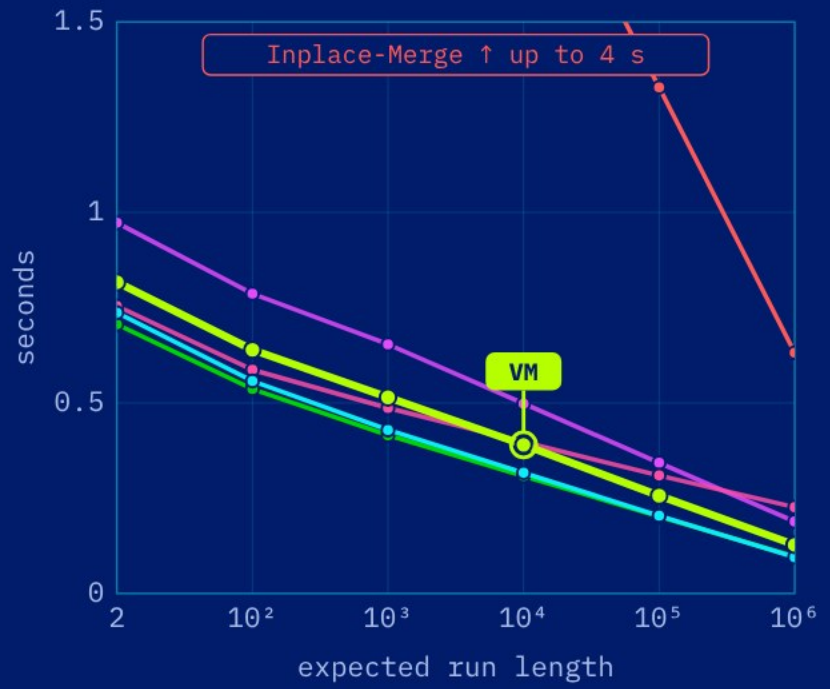
Inplace-Merge — O(lg n) · ≈ 0

Virtual-Memory — 1 MB · 2.8 %

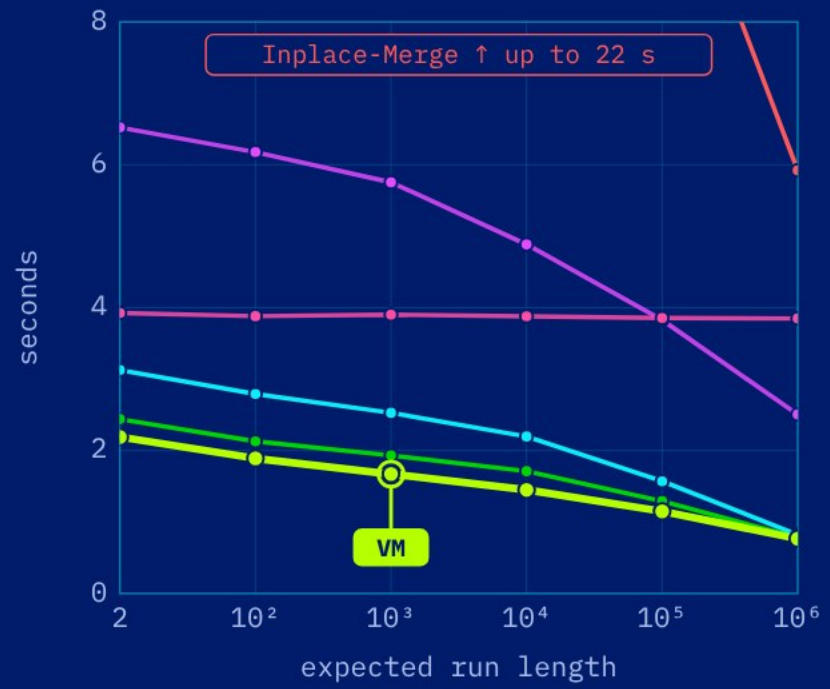
Pingpong — 36 MB · 100 %

120-byte records (1 GB input):
6 MB instead of 543 MB.

running time · int



running time · 120-byte records



CPython-style Powersort · Inplace-Merge Powersort · Virtual-Memory Powersort
Pingpong Powersort · std::stable_sort · Wikisort

- For reference: std::stable_sort (not run-adaptive) and Wikisort (in-place) are slower.

MORE

Paper, code & interactive demos

paper page

Full paper, code and data.

wild-inter.net/publications/moltmann-nakajima-wild-2026

interactive demo

Step through Virtual-Memory Powersort on your own inputs — Pingpong Powersort, too.

power-sort.github.io/virtual-memory-powersort

→ **Come by the poster** for Pingpong Powersort and all four data types.

F. Moltmann, T.-V. Nakajima, S. Wild: Virtual-Memory Powersort. ESA 2026, LIPIcs 388, 14:1–14:17 · doi 10.4230/LIPIcs.ESA.2026.14 · code git.sr.ht/~tamionv/vm-powersort