

Segovia: a chunked, GIL-released, bounded-memory streaming engine for near-real-time multichannel electrophysiology preprocessing

Felipe Carvajal Brown

Independent Researcher

fcarvajalbrown@gmail.com ORCID: 0000-0002-8300-7587

Abstract

High-density extracellular electrophysiology probes (Neuropixels) acquire 384–768 channels at 30 kHz, producing about 22 MB/s per probe. Preprocessing this stream for near-real-time use, such as closed-loop stimulation or online brain–machine-interface decoding, means each incoming chunk of samples must be processed inside its acquisition period while peak memory stays bounded and independent of how long the recording runs. Batch tools driven one chunk at a time re-read the filter-margin neighbourhood on every call and give no analytical memory bound. We describe Segovia, an open-source Rust library (AGPL-3.0) with Python bindings (PyO3) built for this regime. Its core abstraction is the `ChunkSource` trait: an iterator over fixed-size `i16` chunk buffers consumed by a Rayon-parallelized bandpass-filter, common-median-reference, and ZCA-whitening chain. The Python Global Interpreter Lock is released for the duration of each Rust computation, and peak memory is bounded analytically by chunk size rather than recording length. We evaluate Segovia on a real International Brain Laboratory Neuropixels AP-band recording (385 channels, mtscomp-compressed) and on a built-in streaming synthetic simulator (`SyntheticEphysReader`), using a replay-at-acquisition-rate harness that measures per-chunk latency and deadline adherence without hardware. Streaming the full 55.8-minute recording at a 300 ms chunk budget, Segovia meets its real-time deadline on 99.7% of chunks at 0.21 GB peak memory, against 94.7% at 0.41 GB for an equivalent SpikeInterface online-streaming configuration; the larger separations are in peak memory (2×), maximum latency (334 vs 932 ms), and jitter (39 vs 60 ms). In the cold-start window the adherence gap is wider (100% vs 69.5% at the same budget) because SpikeInterface pays its per-chunk setup cost most heavily before the run warms up. Segovia is available at <https://github.com/fcarvajalbrown/Segovia> and via `pip install segovia`.

1 Introduction

Neuropixels probes (Jun et al., 2017) have made large-scale, high-density extracellular electrophysiology routine. A single probe acquires 384 channels at 30 kHz for about 22 MB/s, and multi-probe rigs multiply that by four to eight. A one-to-two-hour session produces tens of gigabytes of raw data. Before spike sorting can begin, this stream passes through a preprocessing chain: bandpass filtering to isolate action-potential frequencies (300–6000 Hz), common-median referencing (CMR) to suppress common-mode noise, and whitening to decorrelate channels. In the standard offline pipeline (SpikeInterface (Buccino et al., 2020), MountainSort), this step runs after recording finishes and is tuned for throughput on a fixed dataset.

Near-real-time applications work under different constraints. Closed-loop optogenetic stimulation must detect a neural event and respond within tens to hundreds of milliseconds, inside the same chunk of samples the event occurred in. Online brain-machine-interface decoders must emit decoded signals with bounded latency. Hardware-in-the-loop experiments must synchronize preprocessing output with external devices at acquisition rate. These uses need two guarantees that batch pipelines do not offer:

1. **Real-time deadline adherence.** Each chunk of N samples must be preprocessed within N/f_s seconds, its chunk period. Missing this deadline stalls the application.
2. **Bounded, file-size-independent peak memory.** A closed-loop system may run for hours or days, so it cannot exhaust RAM as the recording grows.

Drive a batch tool such as SpikeInterface one chunk at a time through sequential `get_traces` calls, the natural online adaptation, and each call re-reads and re-decodes the filter-margin neighbourhood (the look-ahead and look-behind samples the filter needs). Whitening-matrix estimation may also be repeated or anchored to cached state in ways that were never meant for streaming. Memory then grows with the number of processed chunks or with the mapping of the whole recording, not with chunk size alone. Segovia is a purpose-built streaming engine that meets both constraints by construction.

2 Related work

Batch preprocessing frameworks. SpikeInterface (Buccino et al., 2020) gives a unified Python API for spike sorting that includes the standard preprocessing chain. Its `ChunkRecordingExecutor` spreads processing across chunks with a thread or process pool, tuned for bulk throughput. It does not target per-chunk latency or an analytical memory bound. MountainSort (Chung et al., 2017) follows a similar offline model. Kilosort4 (Pachitariu et al., 2024), the current state-of-the-art GPU spike sorter, works on already-preprocessed recordings rather than raw streams. None of these target the online, one-chunk-at-a-time regime.

Streaming and closed-loop frameworks. `improv` (Draeos et al., 2025) is a Python framework for closed-loop calcium-imaging experiments, benchmarked by replaying prerecorded data at acquisition rate rather than against live hardware. BRAND (Ali et al., 2024) is a modular platform for closed-loop experiments with deep-network models, reaching sub-600 μ s latency over asynchronous Redis-based messaging. RT-Sort (van der Molen et al., 2024) does real-time spike detection and sorting with 7.5 ms latency on recorded ground-truth datasets. Together these establish replay-at-acquisition-rate as an accepted way to evaluate near-real-time systems without a live rig (Hoefer and Belli, 2015).

Simulators. MEArec (Buccino and Einevoll, 2021) generates synthetic extracellular recordings with ground-truth spike labels from biophysical neuron models. Segovia’s built-in simulator borrows the MEArec template idea (Ricker waveform, point-source spatial decay) but emits data one chunk at a time as a bounded-memory generator, never materializing the full recording on disk.

Rust in scientific computing. The PyO3 ecosystem (PyO3 contributors, 2024) lets Rust libraries expose Python APIs while releasing the GIL during computation, pairing C-level speed with Python usability. Rayon (rayon contributors, 2024) supplies data-parallel iterators over chunks through a work-stealing thread pool, with no subprocess overhead and no pickle-based inter-process communication.

3 Architecture

3.1 The ChunkSource trait

Segovia’s central abstraction is the **ChunkSource** trait, an iterator over fixed-size ($\text{samples} \times \text{channels}$) `i16` buffers. Each `next()` call returns exactly $\text{chunk_samples} \times n_{\text{channels}}$ samples in row-major ($\text{samples} \times \text{channels}$) order, the layout downstream spike sorters expect. Three production implementations ship:

- **SpikeGlxReader** reads SpikeGLX `.bin + .meta` pairs through memory-mapped I/O. Chunks are sliced from the memory map with zero copy and no decompression.
- **ZarrReader** reads chunked Zarr arrays (gzip, zstd, Blosc codecs) through the `zarrs` crate, aligning Zarr chunk boundaries to the requested chunk size where possible.
- **CbinReader** reads mtscomp-compressed IBL `.cbin` recordings. Each chunk is decompressed on its own through a positioned read with `flate2` (zlib). This is the format used for the real-data benchmark in Section 5.1.

3.2 The preprocessing chain

A **ChunkSource** is consumed by `preprocess(chunk_source, config)`, which chains three operations.

Bandpass filter. A 5th-order Butterworth filter with zero-phase response (`sosfiltfilt`) is applied to each chunk. Zero-phase filtering needs a look-ahead (`margin`) of future samples for the backward pass, so Segovia fetches `margin` samples past the current chunk boundary and drops them from the output, keeping the output time-aligned. This adds a bounded look-ahead latency of margin/f_s (50 ms at the default `margin = 1500` samples, $f_s = 30\,000$ Hz). A causal single-pass mode that removes this look-ahead is planned but not yet built.

Common median reference (CMR). The median across all channels is subtracted sample by sample. This suppresses motion artefacts and common-mode electrical noise without a reference electrode. It runs after bandpass filtering on each chunk on its own.

Global ZCA whitening. The whitening matrix is estimated from the first `n_calib` samples (default 60 000; 2 s at 30 kHz), then stored and reused for every later chunk. Whitening runs in `f32` to avoid integer overflow, and the output is `f32`.

Parallelism and GIL release. The chain is Rayon-parallelized across time-chunks. When `batch > 1`, several chunks run concurrently through Rayon’s work-stealing pool. Cross-chunk filter state (the IIR initial conditions for the forward and backward passes) is carried deterministically: the state for chunk k is seeded from the last `margin` samples of chunk $k - 1$, so the output is identical for any thread count. The Python GIL is released through PyO3’s `allow_threads` for the whole Rust computation, so Python threads blocked on I/O or other work are not held.

Memory bound. Peak memory is bounded by

$$M = \text{batch} \times (\text{chunk_samples} + 2 \times \text{margin}) \times n_{\text{channels}} \times \text{sizeof}(\text{f32}),$$

which does not depend on recording length. At `batch = 1`, `chunk = 9000`, `margin = 1500`, $n_{\text{channels}} = 384$, the bound is about 0.08 GB. Observed peak RSS on real data runs higher (0.18–

0.49 GB) because of the Python runtime, the decompressor, and the whitening matrix, but it still scales only with chunk size, not with recording length.

3.3 Built-in streaming simulator

`SyntheticEphysReader` is a `ChunkSource` that emits arbitrarily long synthetic multi-channel ephys streams without touching a file. Each chunk is generated on demand from this model:

- **Spike templates.** Each unit has a Ricker (Mexican-hat) temporal waveform of amplitude A convolved with a point-source spatial decay $V(r) = A \times d_{\perp}/r$, where r is the probe-to-source distance and $d_{\perp}$ is the perpendicular component. Waveforms are precomputed at initialisation and stamped into the chunk buffer at spike times.
- **Spike times.** Each unit fires on its own as a Poisson process with rate λ .
- **Noise.** Additive white Gaussian noise of standard deviation σ is added to all channels.
- **PRNG.** A SplitMix64 seed generator feeds per-unit xoshiro256++ generators. The PRNG is deterministic across platforms and chunk sizes: split the recording into different chunk sizes and the output is bit-identical.

The simulator’s footprint is one chunk buffer plus the (small, bounded) template bank. It serves two roles: a standalone benchmark source for systems metrics (Section 5.2), and the basis for `ground_truth()`, which returns (sample_index, unit_id, peak_channel) tuples for MEArec-style spike-sorter accuracy evaluation.

4 Evaluation methodology

We follow the replay-at-acquisition-rate method: prerecorded or synthetic data are streamed from disk, or generated, at the true 30 kHz sampling rate, and per-chunk end-to-end latency is measured with nanosecond-precision monotonic clocks (`perf_counter_ns`). A chunk meets its real-time deadline when its compute latency is at most the chunk period ($\text{chunk_samples}/f_s$). Deadline adherence is the fraction of chunks satisfying that bound. The first three chunks of each run are dropped as warm-up.

Metrics reported: per-chunk latency mean, standard deviation (jitter), p95, p99, and max; deadline adherence; peak whole-process RSS (sampled every 20 ms in-process); and sustained throughput (total output bytes divided by total elapsed wall time).

SpikeInterface baseline. SI runs in a separate venv (`.venv-si`, `spikeinterface==0.102.3`) in the same process, driven by sequential `get_traces(start_frame, end_frame)` calls with `n_jobs = 1`, the online analog of Segovia’s `batch = 1`. The filter margin is matched (`bandpass_filter(margin_ms=50.0)`). Whitening uses `mode="global"` with random calibration chunks (the SI default); Segovia uses the first 60 000 samples. Both differences touch only warm-up and are excluded by the three-chunk discard.

Machine: Windows, 8 physical / 16 logical cores, 7.8 GB RAM.

5 Results

5.1 Real IBL AP-band recording

Source: `_spikeglx_ephysData_g0_t0.imec0.ap.cbin` from the IBL reproducible ephys dataset (International Brain Laboratory et al., 2025) (mtscmp-compressed, 385 channels including sync, 30 kHz).

Full length, 300 ms budget (steady state, 55.8 min, 11,167 chunks). This is the representative sustained-streaming measurement and the headline online result (Table 1).

Table 1: Full-length steady-state streaming, 300 ms chunk budget, real IBL recording.

Engine	Mean (ms)	p95 (ms)	p99 (ms)	Max (ms)	Jitter (ms)	Adher.	Peak RSS	Thpt (MB/s)
Segovia	179.2	256.3	277.0	334.5	38.6	99.7%	0.21 GB	38.1
SI online	205.3	301.4	355.0	932.0	60.5	94.7%	0.41 GB	33.3

Segovia leads on every axis. The largest separations are peak memory (2 $\times$), maximum latency (334 vs 932 ms), p99 (277 vs 355 ms), and jitter (38.6 vs 60.5 ms). The deadline-adherence lead is real but narrow at steady state, 99.7% against 94.7%.

Cold-start, first 60 s (1,800,000 samples). The per-chunk sweep in Table 2 covers the cold window: first reads, cold page cache, and library warm-up beyond the three-chunk discard, where SpikeInterface’s per-chunk `get_traces` setup cost is highest and the adherence gap is widest. That gap narrows to the steady-state figures above over the full recording, so these numbers describe cold-start rather than sustained operation. Only the 300 ms leg has a full-length counterpart in Table 1; the 100 ms and 1000 ms rows here remain 60 s cold-start measurements.

Table 2: Cold-start per-chunk sweep, first 60 s of the real IBL recording. 597 / 197 / 57 chunks measured per row.

Chunk	Engine	Mean (ms)	p95 (ms)	p99 (ms)	Max (ms)	Jitter (ms)	Adher.	Peak RSS	Thpt (MB/s)
100 ms	Segovia	92.9	118.4	122.0	127.9	15.4	74.2%	0.21 GB	24.7
100 ms	SI online	112.0	250.0	275.2	302.8	48.5	64.2%	0.46 GB	20.5
300 ms	Segovia	194.5	250.1	256.4	292.5	34.7	100%	0.28 GB	35.2
300 ms	SI online	245.8	355.7	365.7	407.1	67.8	69.5%	0.52 GB	27.9
1000 ms	Segovia	617.3	692.3	705.9	707.9	77.5	100%	0.49 GB	37.4
1000 ms	SI online	786.0	831.3	947.5	1036.6	42.9	98.2%	0.74 GB	29.1

In this cold window at 300 ms, Segovia reaches 100% adherence at 0.28 GB while SpikeInterface online reaches 69.5% at 0.52 GB, a wider gap than the 99.7% vs 94.7% at steady state, because SI pays its per-chunk `get_traces` setup and margin re-decode most heavily before the run warms up (each call re-reads and re-decodes the 50 ms filter-margin neighbourhood, with no cross-chunk pipelining; SI cold p99 is 366 ms against Segovia’s 256 ms). At 100 ms both engines fall short of 100% adherence because the mtscomp zlib decode is memory-bandwidth bound (established in the project’s ADR 0013): the Rust compute meets the deadline but the decode does not. Peak RSS tracks only chunk size on real data, matching the analytical bound, and Segovia’s RSS sits below SI’s at every chunk size.

5.2 Synthetic recordings (Segovia-only baseline)

Source: `SyntheticEphysReader`, 384 channels, 60 s, 30 kHz, 20 units, 5 Hz Poisson firing, 10 μ V noise, seed 0 (Table 3).

Table 3: Synthetic uncompressed stream, Segovia only.

Chunk	Mean (ms)	p95 (ms)	p99 (ms)	Max (ms)	Jitter (ms)	Adher.	Peak RSS	Thpt (MB/s)
100 ms	61.3	68.4	73.7	83.1	3.6	100%	0.15 GB	37.2
300 ms	147.9	163.9	167.4	178.1	8.6	100%	0.23 GB	45.5
1000 ms	617.8	650.9	664.9	672.8	18.6	100%	0.50 GB	36.2

On synthetic uncompressed streams Segovia holds 100% deadline adherence at every chunk size, with much lower jitter than on real compressed data (3.6 ms against 15.4 ms at 100 ms chunks). That lower jitter comes from the absence of zlib decode variance. Throughput clears 22 MB/s in every configuration.

5.3 Batch throughput

The online comparison above is separate from batch throughput, where `SpikeInterface`’s parallel `ChunkRecordingExecutor` ($n_jobs = N$) runs its C/MKL kernels across all cores, the regime SI was designed for. We reran this on the full 55.8-minute (29 GB compressed) real IBL recording, with Segovia at a pinned batch of 4 (the throughput/memory optimum on this host, ADR 0017; the footprint is about $0.17 \text{ GB} \times \text{batch} + 0.5 \text{ GB}$), in Table 4.

Table 4: Full-scale batch throughput, full 55.8-minute real IBL recording.

Engine	Wall time	Peak RSS
Segovia (batch 4)	806 s	1.19 GB
SpikeInterface (thread pool, $n_jobs = 8$)	923 s	2.18 GB
SpikeInterface (process pool, $n_jobs = 8$)	1022 s	4.42 GB

At this pinned batch Segovia used less peak memory than both SI executor modes and finished in less wall time, in a single run. The memory advantage is the robust, deterministic finding, confirmed file-size-independent at full scale: batch-1 peak moved only 0.9% from a ten-minute slice to the full recording. The wall-time margin is a single-machine, single-run measurement and should be replicated with confidence intervals across machines before being read as a general speed claim. An earlier report that Segovia only ties SI in batch (ADR 0013) was taken at the unpinned auto default, which oversubscribes memory-bandwidth-bound work on many-core hosts (ADR 0017). We verified output equivalence: without whitening, Segovia matches SI to 0.0035% (bandpass + CMR), and the whitened-run difference comes from SI’s random-subset whitening calibration against Segovia’s deterministic first-60,000-sample calibration, not from a computational error.

6 Discussion

Online versus batch. The results draw a clear line: batch tools driven one chunk at a time pay a per-chunk overhead that a purpose-built streaming engine avoids. This follows from `SpikeInterface`’s

design goal rather than any flaw in it. Closed-loop applications that need sub-300 ms preprocessing latency with bounded memory should reach for a streaming-first tool; batch tools stay the right choice for offline analysis.

Non-causal filter and look-ahead. The zero-phase Butterworth filter introduces a 50 ms look-ahead (the margin/f_s term), so true end-to-end latency from sample acquisition to preprocessed output is $\text{chunk_period} + 50 \text{ ms} + \text{compute_latency}$, not compute latency alone. At 300 ms chunks that is about 400–450 ms total. A causal single-pass mode would remove the 50 ms look-ahead at the cost of phase distortion, and is future work. Every latency figure in this paper is compute latency; the look-ahead is reported separately by the harness.

Synthetic-data limits. The synthetic simulator does not reproduce the exact noise statistics of real recordings, a documented limit of the MEArec-style approach. The metrics reported here (latency, memory, deadline adherence) are systems metrics that depend on data shape and scale rather than biological fidelity, so the synthetic results support the claims made. We keep the real IBL run for external validity, and it exposes the real decoding bottleneck (mtscomp zlib) that synthetic uncompressed data does not.

Single-machine measurements. All benchmarks run on one machine (Windows, 8 physical cores, 7.8 GB RAM), so performance on other hardware will differ. The analytical memory bound $M = \text{batch} \times (\text{chunk} + 2 \times \text{margin}) \times \text{channels} \times \text{sizeof}(\text{f32})$ is machine-independent; the latency and throughput figures are not.

Future work. A causal single-pass filter mode to remove the 50 ms look-ahead, and parallelized mtscomp decompression to lift the 100 ms budget limit on compressed data.

7 Conclusion

Segovia is a streaming, bounded-memory preprocessing engine for Neuropixels-scale electrophysiology that satisfies two hard constraints for near-real-time work, real-time deadline adherence and file-size-independent memory, which batch tools do not offer in the online regime. Streaming the full 55.8-minute real IBL recording at a 300 ms chunk budget, Segovia meets its deadline on 99.7% of chunks at 0.21 GB against 94.7% at 0.41 GB for SpikeInterface online, and it holds larger margins in peak memory, maximum latency, and jitter. The engine ships as `pip install segovia` (Python, PyPI) and `cargo add segovia` (Rust, crates.io) under AGPL-3.0-or-later.

Acknowledgements

The name Segovia honors Claudio Segovia (1983–2009), a friend who died of leukemia at 26.

References

Yahia H. Ali, Kevin Bodkin, Mattia Rigotti-Thompson, Kushant Patel, Nicholas S. Card, Bareesh Bhaduri, Samuel R. Nason-Tomaszewski, Domenick M. Mifsud, Xianda Hou, Claire Nicolas, Shane Allcroft, Leigh R. Hochberg, Nicholas Au Yong, Sergey D. Stavisky, Lee E. Miller, David M. Brandman, and Chethan Pandarinath. BRAND: a platform for closed-loop experiments with deep

- network models. *Journal of Neural Engineering*, 21(2):026046, 2024. doi: 10.1088/1741-2552/ad3b3a.
- Alessio P. Buccino, Cole L. Hurwitz, Samuel Garcia, Jeremy Magland, Joshua H. Siegle, Roger Hurwitz, and Matthias H. Hennig. SpikeInterface, a unified framework for spike sorting. *eLife*, 9:e61834, 2020. doi: 10.7554/eLife.61834.
- Alessio Paolo Buccino and Gaute Tomas Einevoll. MEArec: A fast and customizable testbench simulator for ground-truth extracellular spiking activity. *Neuroinformatics*, 19(1):185–204, 2021. doi: 10.1007/s12021-020-09467-7.
- Jason E. Chung, Jeremy F. Magland, Alex H. Barnett, Vanessa M. Tolosa, Angela C. Tooker, Kye Y. Lee, Kedar G. Shah, Sarah H. Felix, Loren M. Frank, and Leslie F. Greengard. A fully automated approach to spike sorting. *Neuron*, 95(6):1381–1394.e6, 2017. doi: 10.1016/j.neuron.2017.08.030.
- Anne Draelos, Matthew D. Loring, Maxim Nikitchenko, Chaichontat Sriworarat, Pranjal Gupta, Daniel Y. Sprague, Eftychios Pnevmatikakis, Andrea Giovannucci, Tyler Benster, Karl Deisseroth, John M. Pearson, and Eva A. Naumann. A software platform for real-time and adaptive neuroscience experiments. *Nature Communications*, 16:9909, 2025. doi: 10.1038/s41467-025-64856-3.
- Torsten Hoefler and Roberto Belli. Scientific benchmarking of parallel computing systems: twelve ways to tell the masses when reporting performance results. In *SC ’15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 73:1–73:12, 2015. doi: 10.1145/2807591.2807644.
- International Brain Laboratory, Kush Banga, Julius Benson, Jai Bhagat, Dan Biderman, Daniel Birman, Niccolò Bonacchi, Sebastian A. Bruijns, Kelly Buchanan, Robert A.A. Campbell, Matteo Carandini, Gaelle A. Chapuis, Anne K. Churchland, M. Felicia Davatolhagh, Hyun Dong Lee, Mayo Faulkner, Berk Gerçek, Fei Hu, Julia Huntenburg, Cole Lincoln Hurwitz, Anup Khanal, Christopher Krasniak, Petrina Lau, Christopher Langfield, Nancy Mackenzie, Guido T. Meijer, Nathaniel J. Miska, Zeinab Mohammadi, Jean-Paul Noel, Liam Paninski, Alejandro Pan-Vazquez, Cyrille Rossant, Noam Roth, Michael Schartner, Karolina Z. Socha, Nicholas A. Steinmetz, Karel Svoboda, Marsa Taheri, Anne E. Urai, Shuqi Wang, Miles Wells, Steven J. West, Matthew R. Whiteway, Olivier Winter, Ilana B. Witten, and Yizi Zhang. Reproducibility of in vivo electrophysiological measurements in mice. *eLife*, 13:RP100840, 2025. doi: 10.7554/eLife.100840.3.
- James J. Jun, Nicholas A. Steinmetz, Joshua H. Siegle, Daniel J. Denman, Marius Bauza, Brian Barbarits, Albert K. Lee, Costas A. Anastassiou, Alexandru Andrei, Çağatay Aydın, Mladen Barbic, Timothy J. Blanche, Vincent Bonin, João Couto, Barundeb Dutta, Sergey L. Gratiy, Diego A. Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina Mora Lopez, Catalin Mitelut, Silke Musa, Michael Okun, Marius Pachitariu, Jan Putzeys, P. Dylan Rich, Cyrille Rossant, Wei-Lung Sun, Karel Svoboda, Matteo Carandini, Kenneth D. Harris, Christof Koch, John O’Keefe, and Timothy D. Harris. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017. doi: 10.1038/nature24636.
- Marius Pachitariu, Shashwat Sridhar, Jacob Pennington, and Carsen Stringer. Spike sorting with Kilosort4. *Nature Methods*, 21(5):914–921, 2024. doi: 10.1038/s41592-024-02232-7.
- PyO3 contributors. PyO3: Rust bindings for the python interpreter. <https://pyo3.rs>, 2024. Version 0.28.

rayon contributors. Rayon: A data parallelism library for Rust. <https://github.com/rayon-rs/rayon>, 2024.

Tjitse van der Molen, Max Lim, Julian Bartram, Zhuowei Cheng, Ash Robbins, David F. Parks, Linda R. Petzold, Andreas Hierlemann, David Haussler, Paul K. Hansma, Kenneth R. Tovar, and Kenneth S. Kosik. RT-Sort: An action potential propagation-based algorithm for real time spike detection and sorting with millisecond latencies. *PLOS ONE*, 19(12):e0312438, 2024. doi: 10.1371/journal.pone.0312438.