timsRust: A Modular Foundation for Performant timsTOF Data Workflows

Jonathan Krieger[1]; Jonathan Moss[2]; George Rosenberger[3]; Dennis Trede[4]; Sander Willems[5]

[1] Bruker Ltd., Milton, Toronto, ON, Canada
[2] Bruker, Preston, Australia
[3] Bruker Switzerland AG, Fällanden, Switzerland
[4] Bruker Daltonics GmbH & Co KG, Bremen, Germany
[5] Bruker, Kontich, Belgium

Introduction

Trapped ion mobility spectrometry (TIMS) coupled with time of flight (TOF) mass spectrometry (MS) has become an important tool in proteomics. Rather than only filtering in the ion mobility domain, timsTOF instruments provide an additional analytical dimension by continuous separation. This makes more acquisition strategies possible and improves information content, but it also produces larger datasets that require efficient, scalable processing. While several tools are available for working with timsTOF data, many of them lack transparency, flexibility or performance. Moreover, ion mobility resolved data often remains incompatible with existing software tools. Here we present timsRust, a fast, open source, and extensible Rust framework for raw timsTOF data access and (pre-)processing, thereby streamlining integration into other pipelines (figure 1).

Methods

timsRust is an ecosystem consisting of several documented, tested and modular crates that are easy to extend and integrate into existing pipelines as needed (figure 2). Alternatively, it can be used as a single unified façade crate. The framework uses Rust's safety guarantees around concurrency, memory, and typing for performance while simultaneously reducing fatal errors in long cohort wide analyses. Wrappers allow usage of timsRust in Python or C while benefitting from all Rust's advantages. Beyond raw data access, timsRust offers preprocessing steps like centroiding and pseudo MS2 spectrum generation from DIA data, producing data formats compatible with tools like Sage[1] or others with and without ion-mobility awareness.

[1]*Lazear, M. Sage: An Open-Source Tool for Fast Proteomics Searching and Quantification at Scale, Journal of Proteome Research. 22(11), 2023.*

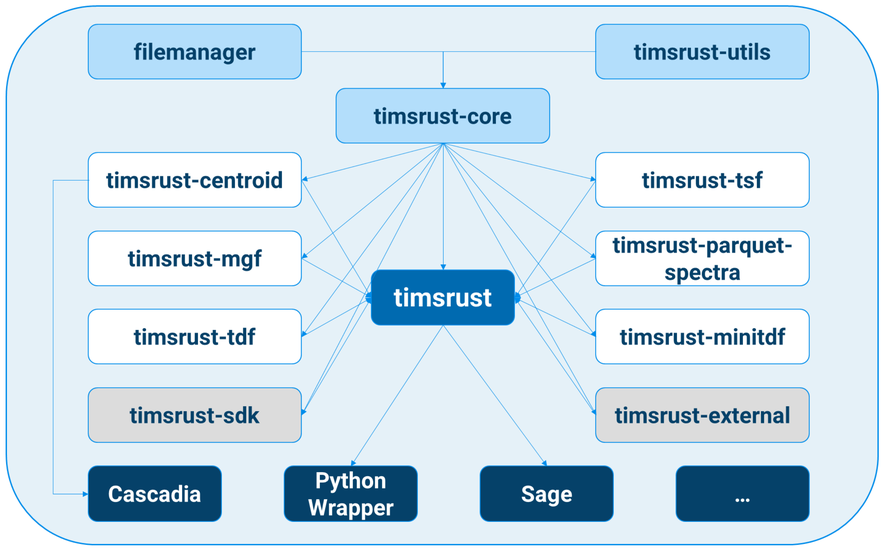

Figure 2: timsRust Architecture. The timsRust ecosystem is organized as a collection of modular Rust crates. Shared types, traits, and utilities are defined in timsRust-core, filemanager, and timsRust-utils (light-blue background). Format-specific crates (white background) each handle reading and conversion for a particular file format, and optionally (pre-)processing. E.g. timsRust-centroid allows centroiding, while equally providing a spectrum reader based on this centroiding. The timsRust-sdk and timsRust-external crates (grey background) are optional dependencies. All crates converge into the timsRust façade crate, which presents a unified API to downstream consumers such as Sage, Cascadia, a Python wrapper, and other tools (dark-blue background).

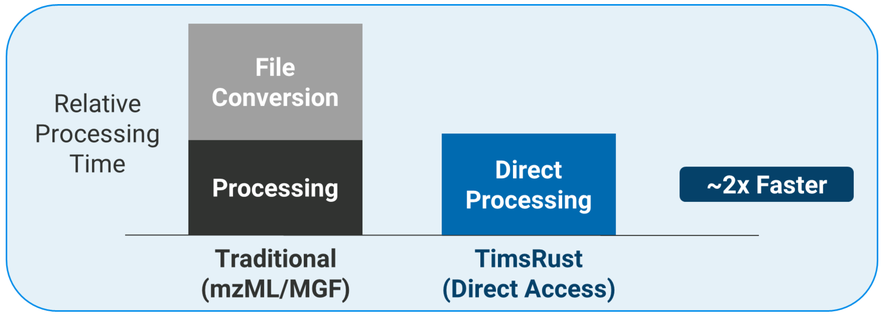

Figure 3: Workflow Speed Comparison. Relative processing time for a traditional conversion-based workflow (via mzML/MGF intermediate files) versus direct raw data access through timsRust. By bypassing file conversion entirely, timsRust eliminates a major bottleneck, substantially reducing total processing time. Bars represent relative durations; absolute timings depend on dataset size, gradient length, and hardware.

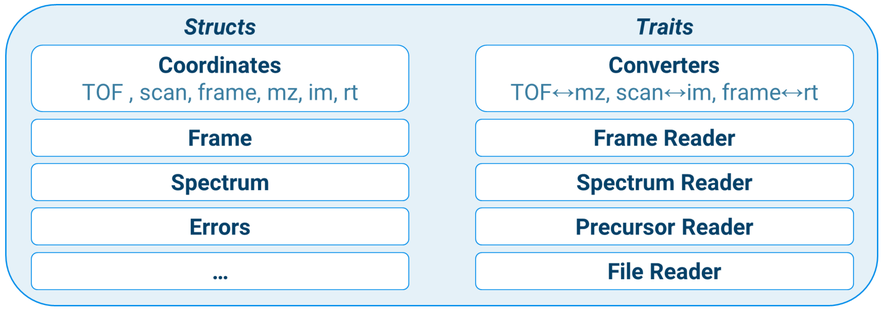

Figure 1: timsRust-core Components. The core crate defines the shared interface for the timsRust ecosystem. It exposes a set of common data structs such as typed coordinates, frames, and spectra. It also provides traits such as converters and readers for these structs. Each format-specific crate (e.g. timsRust-tdf, timsRust-tsf) implements a subset of these traits, while the timsRust façade crate unifies all implementations behind enums, giving downstream users a single entry point regardless of the underlying file format.

Results

Initial benchmarks show that timsRust provides a fast and reliable foundation for timsTOF data analysis across different workflows.

Integrated with the open-source Sage search engine, it reads raw Bruker data directly during peptide-spectrum matching, removing the need for mzML or MGF conversion. Since generating these intermediate files often takes longer than processing them, bypassing conversion immediately reduces overhead. Reading spectra straight from raw data is as fast as reading from converted formats, often even faster, making full workflows roughly twice as fast (figure 3).

In our diaPASEF workflows, the first step is to centroid ions frame-by-frame. Reducing each frame to a set of centroids makes the ion-mobility dimension easier to handle. Afterwards, even simple binning is often enough to treat the data like conventional DIA. This allows peptide-centric tools to analyse the data effectively, regardless of whether they support ion mobility natively.

As a small extension, we link centroided fragments to centroided precursors when their ion mobility and isolation window match, producing pseudo-MS2 spectra that can be analysed directly with Sage. These spectra perform remarkably well, especially on short gradients. On a laptop with 28 virtual cores, generating ~150,000 spectra from a 7-minute HeLa run takes about 15 seconds (~100 µs per spectrum), and the full Sage workflow finishes in about 30 seconds. Centroiding effectively reduces noise and data volume while preserving the information most relevant for peptide-level interpretation (figure 4).

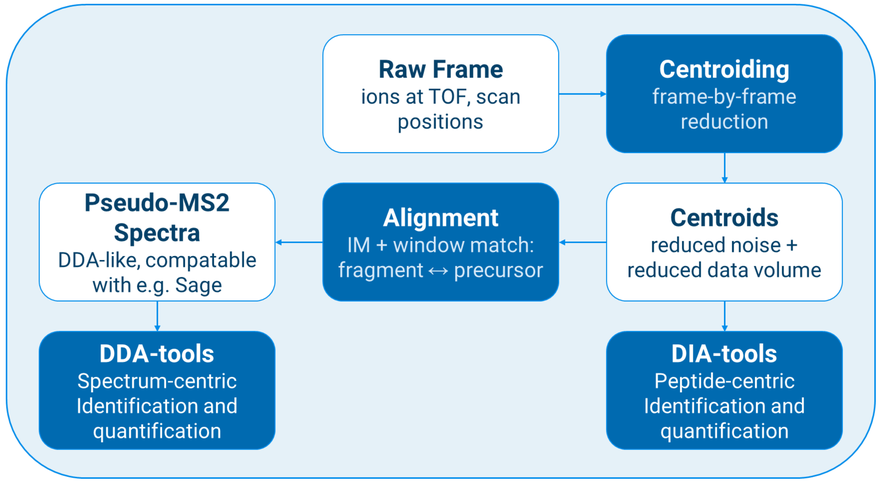

Figure 4: Frame-by-Frame Centroiding. Raw ions are centroided frame by frame, reducing noise and data volume while preserving information relevant for peptide-level interpretation. The centroided data can be processed with conventional DIA tools or alternatively fragments and precursors can be aligned to create pseudo-MS2 spectra for conventional DDA tools.

Summary & Conclusion

A modular, high-performance Rust framework for direct timsTOF data access and integrated preprocessing, enabling efficient and scalable workflows.

timsRust enables both fast DDA and DIA pipelines without specialized hardware. Direct raw access and memory-safe multithreading scale efficiently with CPU cores, reducing processing time and simplifying integration into peptide-centric workflows.

Conflict of Interest Disclosure: all authors are currently employed by Bruker.