polars-bio: High-Performance Python DataFrame Operations for Genomics

How to build composable data management systems at scale?

Marek Wiewiórka

December 05, 2025

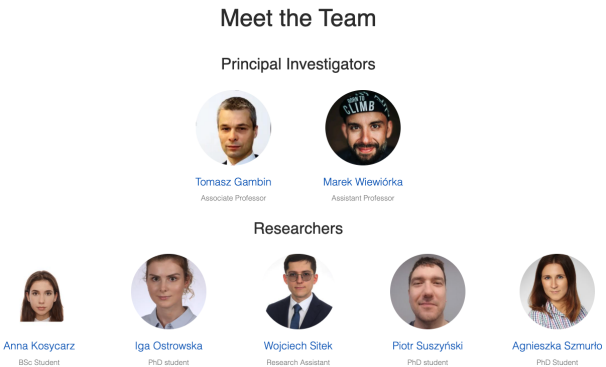
About me

- Assistant Professor[a] at Warsaw University of Technology
- Chief Architect @Xebia Data Poland, 20+ years building data-intensive systems
- distributed and data-intensive systems, artificial intelligence and cloud computing for large scale genomic studies.
- road and gravel bikes enthusiast
- https://marekwiewiorka.org/

[a]Institute of Computer Science

Biodatageeks lab

- Warsaw University of Technology, Faculty of Electronics and Information Technology
- 🔬 current research topics:
  - AI for analyzing biomedical literature
  - Meta-calling for gene fusion detection in RNA-Seq
  - Optimizing RVAS
  - Open genomic data lakehouse
- https://biodatageeks.org/
- https://github.com/biodatageeks/

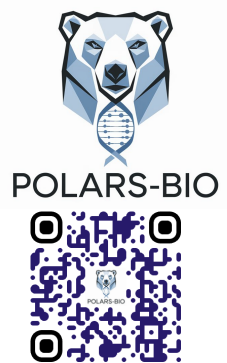


Agenda

1. 🤔 Rationale and motivation for building a domain-specific data management system
2. 🗺️ Composable Data Management System principles
3. 🔬 Deep dive into internals
4. 📊 Benchmarks
5. 🔮 Conclusions

Introduction to polars-bio

- polars-bio is a novel Python DataFrame library for genomics that is *fast* and *memory-efficient*, introduced in 2025, built on top of Polars, Apache DataFusion and Apache Arrow.
- main focus areas:
  - 🧬 genomic interval operations
  - 🚀 scalable data processing and querying
  - 💾 fast I/O for bioinformatics file formats
  - ☁️ cloud storage interoperability
  - 🧱 genomic data lakehouse readiness

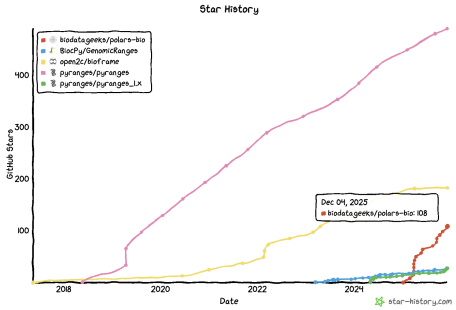


Rationale, History, and Challenges

- 📈 Growing bioinfo dataset sizes vs. increasing capacity of commodity hardware
- ⚖️ Trade-off: scalability of distributed systems (e.g., Apache Spark - Hail, Glow) vs. simplicity and performance of single-node libraries (e.g. DuckDB)
- 🖥️ Single-node solutions: constrained in both performance and scalability
- 🧪 First attempt (2019–2023): SeQuiLa project on top of Apache Spark
- 🔀 Conclusion: towards a *hybrid* approach

Landscape of tools for genomic interval operations in Python

- several widely used libraries exist in this space:
  - Pyranges and new Pyranges1
  - Pybedtools
  - Bioframe
  - GenomicRanges
- employing an *eager*, *in-memory* execution model with Pandas DataFrames/ NumPy arrays
- sweep-line (Bioframe, Pyranges1) or Nested Containment List (Pyranges, GenomicRanges) or genome binning algorithm (Pybetools)
- focus *primarily* on optimizing genomic operations rather than end-to-end processing and IO operations



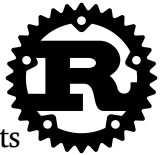
Limitations of Current Approaches to Genomic Interval Processing

Genomic intervals processing is closer to BI/DWH/ETL-style workloads than to numerical computing!

- Relying on libraries (e.g., NumPy) not designed for efficient bioinformatics data handling
- Re-implementing algorithms and reinventing the wheel instead of leveraging mature *query engine*: optimizers, operators and open data standards
- *Parallelism* and *out-of-core* not treated as a first-class concern (limited scalability)
- *Naive* Python implementations (slow, limited scalability)
- Missing *end-to-end optimization* including reading, processing and writing data

Market trends in data systems

- out-of-core (streaming) processing
- single node vectorized engines – e.g. DuckDB, Polars
- *lazy* evaluation and query optimization – e.g. Polars
- open data standards and interoperability, such as Apache Arrow or Apache Iceberg
- composability and reusability, e.g. query parsers, optimizers, query engines, memory and file/table formats
- data lakehouse architecture



The Composable Data Management Systems (CDMS) Manifesto

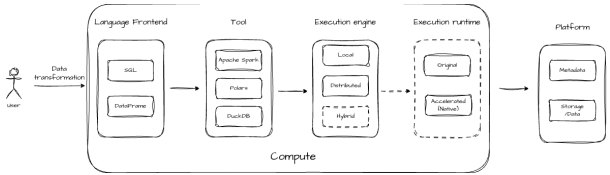
- **Problem**: Data systems are *fragmented*, *duplicated*, hard to maintain
- **Vision**: Break *monoliths* into *modular*, *reusable* components (frontends, Internal Representation, optimizers, execution engines, runtime environments)
- **Why Now**: Already existing *open standards* (Arrow, Parquet, Iceberg)) or implementations (Apache DataFusion, Velox, Apache Spark) to enable *composability*
- **Benefits**: *Faster* innovation, *reduced* engineering effort, consistent user experience

Source: Sourc:https://www.vldb.org/pvldb/vol16/p2679-pedreira.pdf

From Monoliths to Modular Data Stack

Decoupling at different levels (top-down):

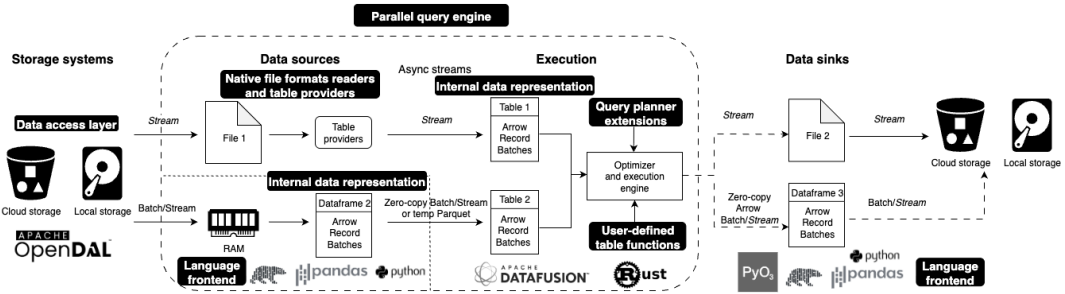
1. Storage vs. Compute vs. Metadata (Data Platform Level)
2. Language Frontend vs. Query Processing (Frontend Level)
3. Query Processing vs. Query Execution Runtime (Engine Level: local/parallel, distributed, hybrid)
4. Original Execution (typically JVM or Python) vs. Native (Rust/C++) High Performance (Runtime Level)



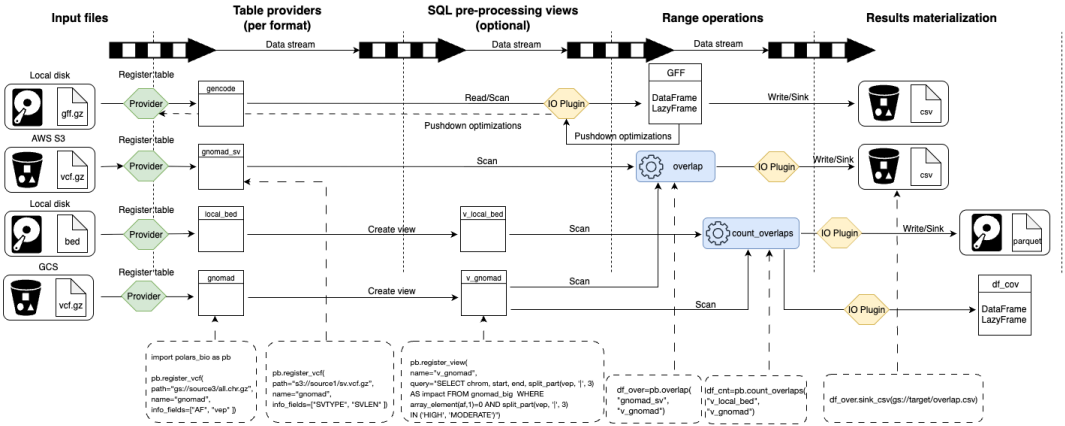
Why polars-bio is different? Composability first!

- Composable and best of breed approach
  - query engine (Apache DataFusion)
  - DataFrame library (Polars)
  - columnar memory format (Apache Arrow)
  - data structure for interval intersection queries (COITrees and Superintervals)
  - bioinformatics file formats (noodles)
  - storage access (OpenDAL)
- Builtin *lazy*, *out-of-core* and *parallel* computational model
- IO layer optimizations for analytical queries, such as *projection* and *predicate* pushdowns

polars-bio high-level architecture

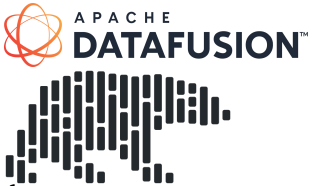


Architecture deep-dive - core components



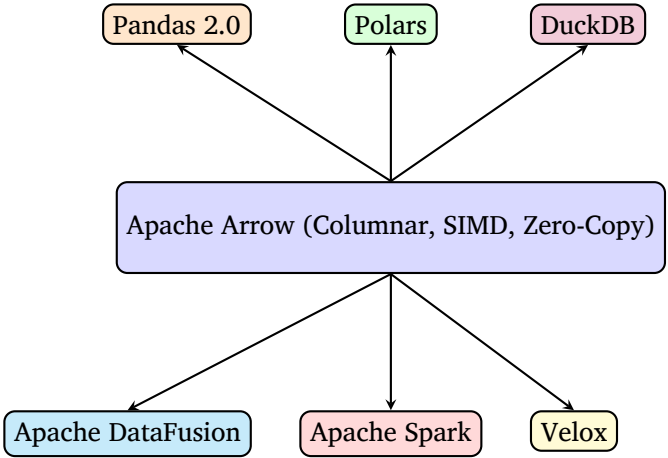
Architecture deep-dive - Polars and Apache DataFusion primer

- Polars and Apache DataFusion exhibit significant similarities, such as Apache Arrow columnar memory model, lazy evaluation and out-of-core computational model, great performance
- different main focuses:
  - Polars – feature-rich end-user DataFrame library
  - DataFusion – extremely extensible query engine for building custom data systems
- do we really need both?
  - Polars' great data wrangling capabilities but hard to extend
  - DataFusion's codebase reusability (e.g. hybrid execution) and more robust abstractions for query and IO optimizations
  - additional integration complexity (e.g. pushdown optimizations, parallelism control)



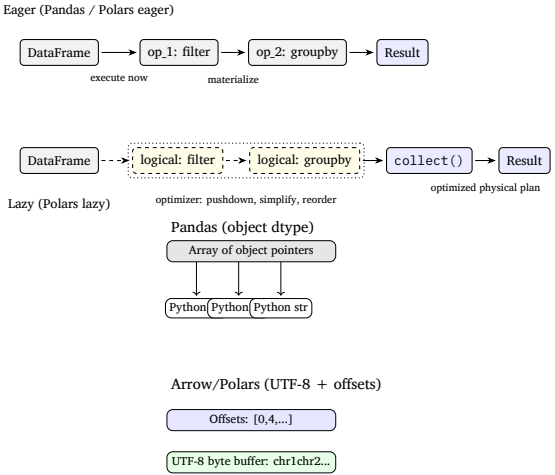
Architecture deep-dive - Apache Arrow

- Standardized columnar memory format – zero-copy sharing
- Vectorized execution: SIMD and CPU cache efficiency
- Cross-language interoperability (e.g. Python and Rust)
- Integration with open standards – Parquet, Iceberg
- Foundation for modern data systems – Polars, Ray, Rapids, Apache Spark



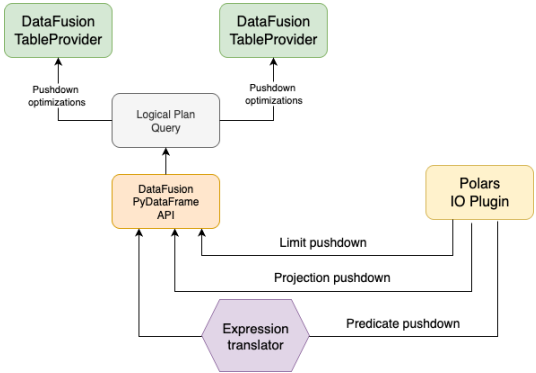
Architecture deep-dive - Polars vs. Pandas

- Execution model: Pandas is eager-only; Polars supports eager *and* lazy.
- Optimization: Pandas has no query optimizer; Polars (lazy) performs projection/predicate pushdown, simplification, reordering.
- Parallelism: Pandas mostly single-threaded (Python/GIL); Polars is multi-threaded (Rust).
- Memory/layout: Pandas uses NumPy blocks; Polars is columnar and Arrow-friendly.
- Out-of-core/streaming: Pandas primarily in-memory; Polars supports streaming/out-of-core in lazy plans.
- String handling: Pandas often stores Python objects (high memory overheads); Polars stores UTF-8 natively with efficient kernels (SIMD).



Architecture deep-dive - Polars IO plugin

- arbitrary function that returns a generator (Iterator) producing pl.DataFrame batches and gets back LazyFrame
- used for both files scanning and interval operations results streaming
- zero-copy and streaming using Arrow RecordBatchStream with DataFusion PyDataFrame
- support for limit, projection and predicate pushdowns (currently only GFF)



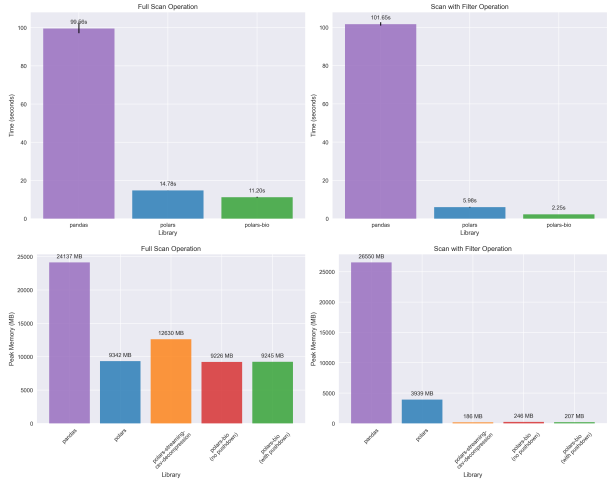
Benchmarking dataset

- AILIst real dataset converted into Parquet format – details
- GFF3 GENCODE release 49

Dataset#	Name	Size(x1000)	Description
0	chainRn4	2,351	Source
1	fBrain	199	Source
2	exons	439	Dataset used in the BEDTools tutorial.
3	chainOrnAna1	1,957	Source
4	chainVicPac2	7,684	Source
5	chainXenTro3Link	50,981	Source
6	chainMonDom5Link	128,187	Source
7	ex-anno	1,194	Dataset contains GenCode annotations with ~1.2 million lines, mixing all types of features.
8	ex-rna	9,945	Dataset contains ~10 million direct-RNA mappings.

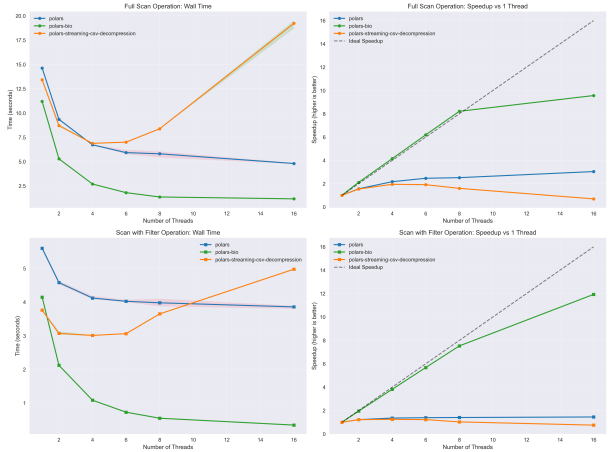
Source: Jianglin Feng , Aakrosh Ratan , Nathan C Sheffield, *Augmented Interval List: a novel data structure for efficient genomic interval search*, Bioinformatics 2019.

File formats: GFF (scan_csv vs polars-bio) – results 1/2



- in full-scans Polars and polars-bio significantly outperform Pandas
- Polars problem with `scan_csv` and compressed files)
- streaming decompression plugin

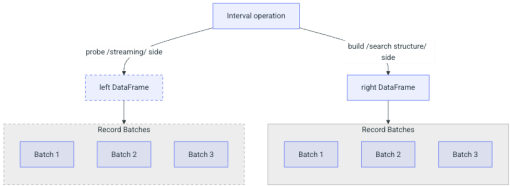
File formats: GFF (scan_csv vs polars-bio) – results 2/2



- polars-bio achieves near-linear scaling up to 8 threads
- Polars and streaming decompression plugin scale poorly

Architecture deep-dive – genomic interval operations 1/2

- inspired by the Hash Join implementation in DataFusion
- the *entire* (coordinates) build side is read into the interval search data structure
- batches from the probe side are *streamed* through and checked against the contents of the search data structure

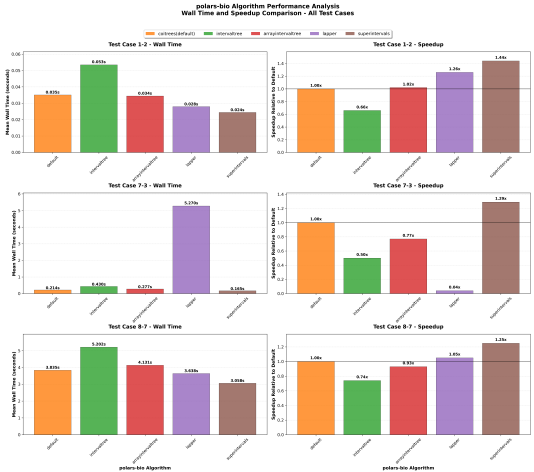


Architecture deep-dive – genomic interval operations 2/2

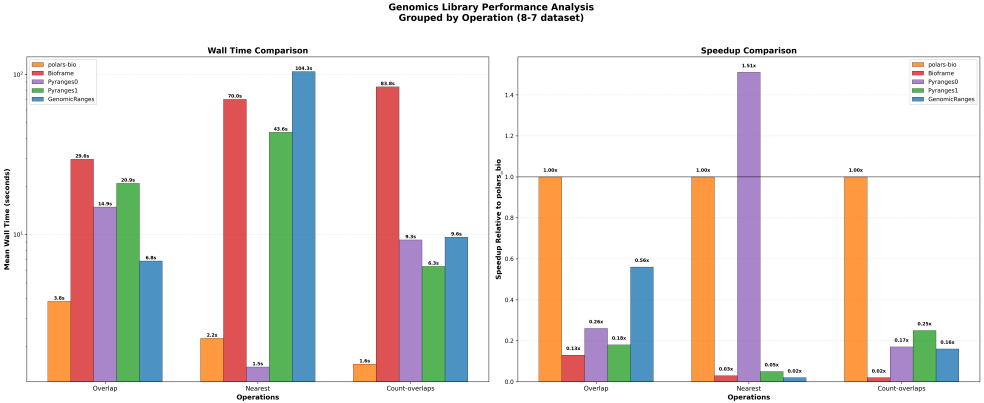
- subproject sequila-native
- custom PhysicalPlanner and PhysicalOptimizerRule for detecting and rewriting generic interval join operation (overlap or nearest)
- User-Defined Table Function (UDTF) for operations, such as coverage or count overlaps
- several data structures available:
  - COITrees
  - IITree
  - AVL-tree
  - rust-lapper
  - Superintervals

Genomic interval operations – structures comparison results 1/5

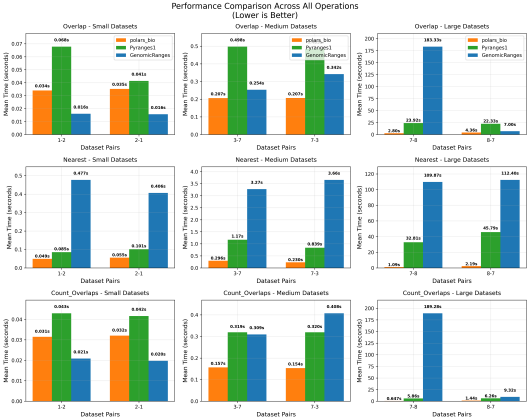
- COITrees (polars-bio default) and Superintervals fastest in all test cases
- configurable in runtime
- more tests using different datasets characteristics needed

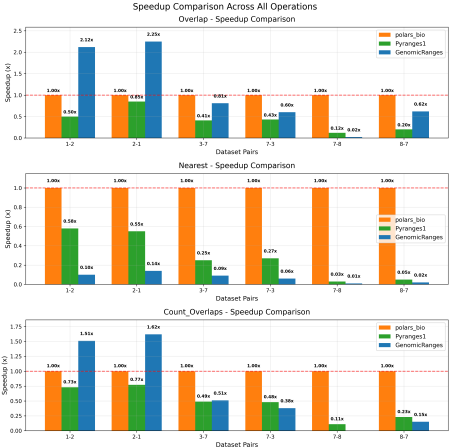


Genomic interval operations – results 2/5

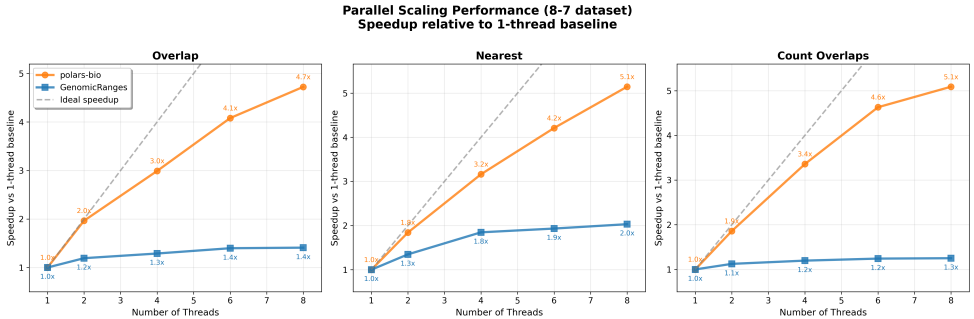


Genomic interval operations – results 3/5

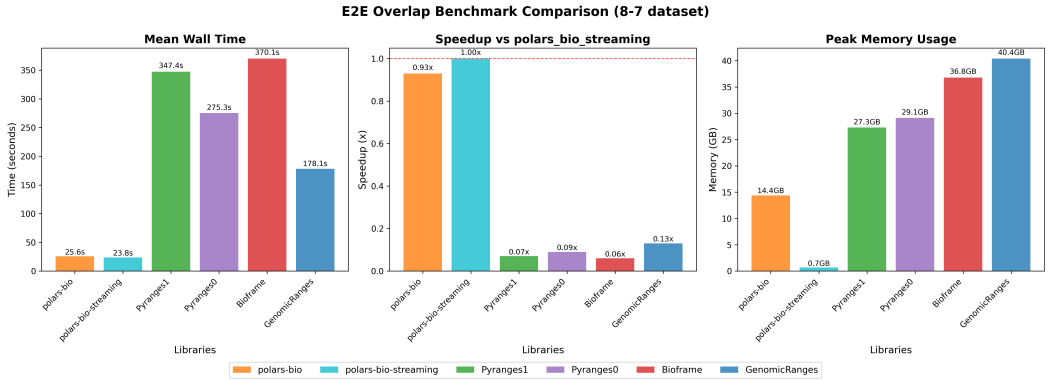




Genomic interval operations – scaling – results 4/5



Genomic interval operations – e2e pipeline – results 5/5



polars-bio research paper

OXFORD ACADEMIC Journals Books

Sign in through your institution

Remember my institution

Bioinformatics

ISCB International Society for Computational Biology

Issues Advance articles Submit Alerts About

Bioinformatics

AI Discovery Assistant

Article Contents

Abstract

Supplementary data

JOURNAL ARTICLE ACCEPTED MANUSCRIPT

polars-bio—fast, scalable and out-of-core operations on large genomic interval datasets

Marek Wiewiórka, Pavel Khamutou, Marek Zbysiński, Tomasz Gambin

Bioinformatics, btaf640, https://doi.org/10.1093/bioinformatics/btaf640

Published: 01 December 2025 **Article history**

PDF Split View Cite Permissions Share

Abstract

Motivation

Genomic studies very often rely on computationally intensive analyses of relationships between features, which are typically represented as intervals along a one-dimensional coordinate system (such as positions on a chromosome). In this context, the Python programming language is extensively used for manipulating and analyzing data stored in a tabular form of rows and columns, called a DataFrame. Pandas is the most widely used Python DataFrame package and has been criticized for inefficiencies and scalability issues, which its modern alternative—Polars—aims to address with a native backend written in the Rust programming language.

CITATIONS 7 VIEWS 0 ALTMETRIC 6

More metrics information

Email alerts

New journal issues

New journal articles

Activity related to this article

Sign up for marketing

Recommended

The Pfam protein families database: embracing AI/ML

https://doi.org/10.1093/bioinformatics/btaf640

Summary

- 🧬 `polars-bio`: a new Python DataFrame library for genomics built upon CDMS principles
- 🚀 Combines Polars, Apache DataFusion, and Apache Arrow for speed and scalability
- 💾 Efficient I/O for popular bioinformatics formats
- ⚖️ Addresses limitations of existing interval processing tools
- 🏗️ Towards a hybrid, lakehouse-ready approach for large-scale genomics

Thank You!

🤝😊

Questions ❓