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

Marek Wiewiórka[1], **Pavel Khamutou, Marek Zbysiński, Tomasz Gambin**

Institute of Computer Science, Warsaw University of Technology

[1]marek.wiewiorka@pw.edu.pl, https://biodatageeks.org/polars-bio/

Background

Genomic studies often rely on computationally intensive analyses of relationships between features, typically represented as intervals along one-dimensional coordinate systems (e.g., chromosome positions). Existing Python genomic interval libraries—such as PyRanges, Bioframe, and PyBedtools—rely on *eager, in-memory execution* models and focus primarily on optimizing genomic operations rather than *end-to-end* processing and I/O.

Methods

To address these challenges, we present *polars-bio*, a Python library that, following *Composable Data Management Systems*(1) principles, combines the strengths of **Apache DataFusion**(2)—an extensible, *columnar, out-of-core, multi-threaded, vectorized* execution engine—**Apache Arrow**, a columnar memory format for efficient data representation and exchange, and the user-friendly, high-performance **Polars**(3) library. To facilitate real-world genomics workflows, *polars-bio* includes fast readers for standard genomic file formats such as **BED**, **GFF**, **VCF**, **BAM**, and **FASTQ**, with *predicate and projection pushdown* optimizations. It also supports popular cloud object storage systems: **AWS S3**, **Google Cloud Storage**, and **Azure Blob Storage**.

Discussion

- **High performance:** Vectorized, columnar lazy execution (Polars+Apache DataFusion) with predicate/projection pushdown, parallel readers, and **zero-copy** data exchange delivers end-to-end **speedups** and **cost efficiency**.
- **Out-of-core cloud-native:** Streaming and partitioned processing of bioinformatics file formats and **genomic range** operations scale beyond RAM, with first-class S3/GCS/Azure support enabling a unified **Genomic Data Lakehouse**.
- **User-friendly reproducible:** Concise Polars **expressions** and **SQL** interoperability lower the barrier from notebook to production.

References

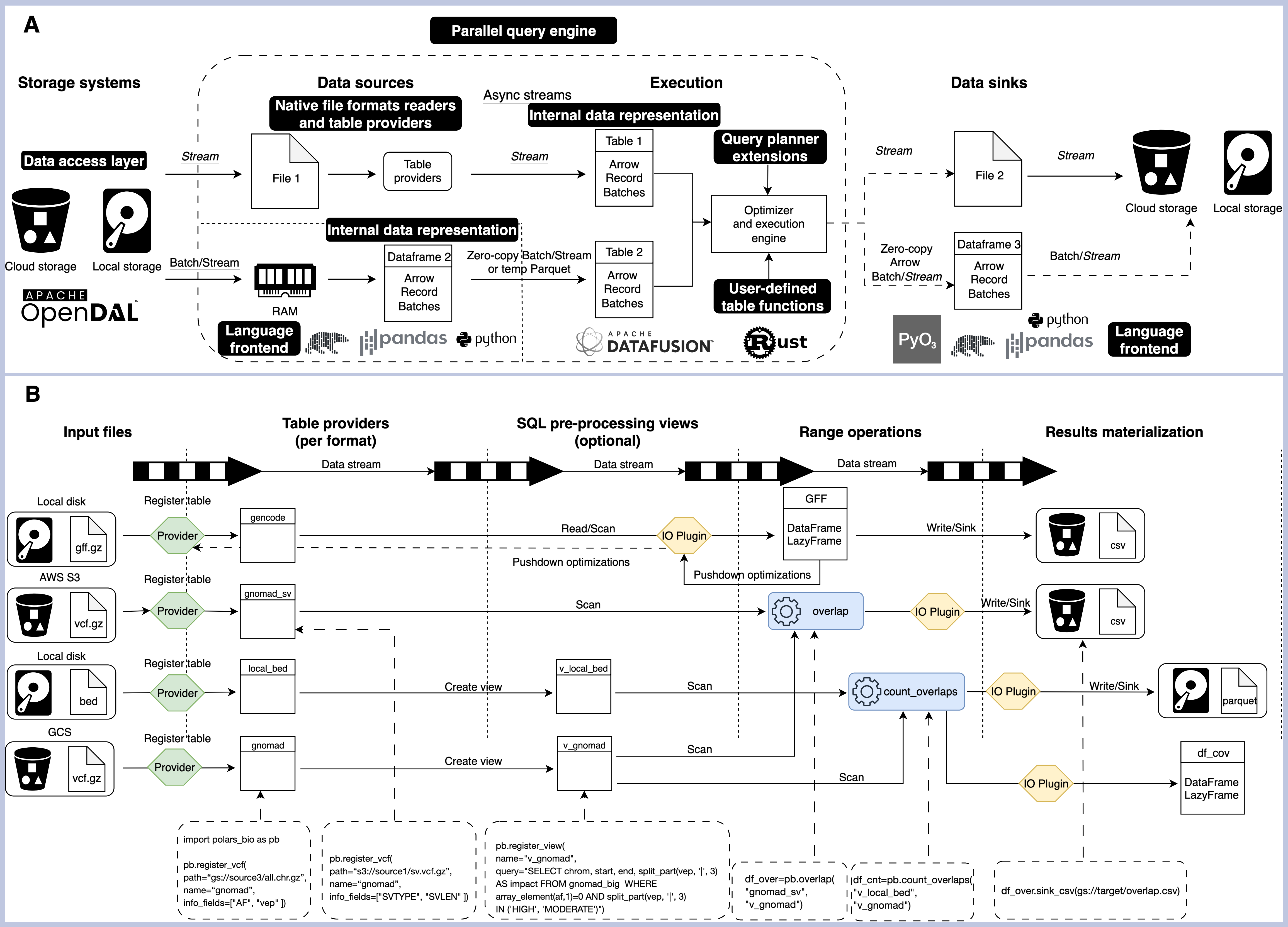
[1] P. Pedreira, O. Erling, K. Karanasos, S. Schneider, W. McKinney, S. R. Valluri, M. Zait, and J. Nadeau, “The Composable Data Management System Manifesto,” *Proceedings of the VLDB Endowment*, vol. 16, pp. 2679–2685, June 2023.

[2] A. Lamb, Y. Shen, D. Heres, J. Chakraborty, M. O. Kabak, L.-C. Hsieh, and C. Sun, “Apache Arrow DataFusion: A Fast, Embeddable, Modular Analytic Query Engine,” SIGMOD/PODS ’24, pp. 5–17, Association for Computing Machinery, 2024.

[3] A. F. Oketunji, “Exploratory Data Analysis with Polars,” Nov. 2024. Version Number: 1.0.0.

[4] J. Feng, A. Ratan, and N. C. Sheffield, “Augmented Interval List: A novel data structure for efficient genomic interval search,” *Bioinformatics*, vol. 35, pp. 4907–4911, 12 2019.

Architecture



```
import polars_bio as pb

pb.register_vcf(
path="gs://source3/all.chr.gz",
name="gnomad",
info_fields=["AF", "vep" ])
```

```
pb.register_vcf(
path="s3://source1/sv.vcf.gz",
name="gnomad",
info_fields=["SVTYPE", "SVLEN" ])
```

```
pb.register_view(
name="v_gnomad",
query="SELECT chrom, start, end, split_part(vep, '|', 3)
AS impact FROM gnomad_big  WHERE
array_element(af,1)=0 AND split_part(vep, '|', 3)
IN ('HIGH', 'MODERATE')")
```

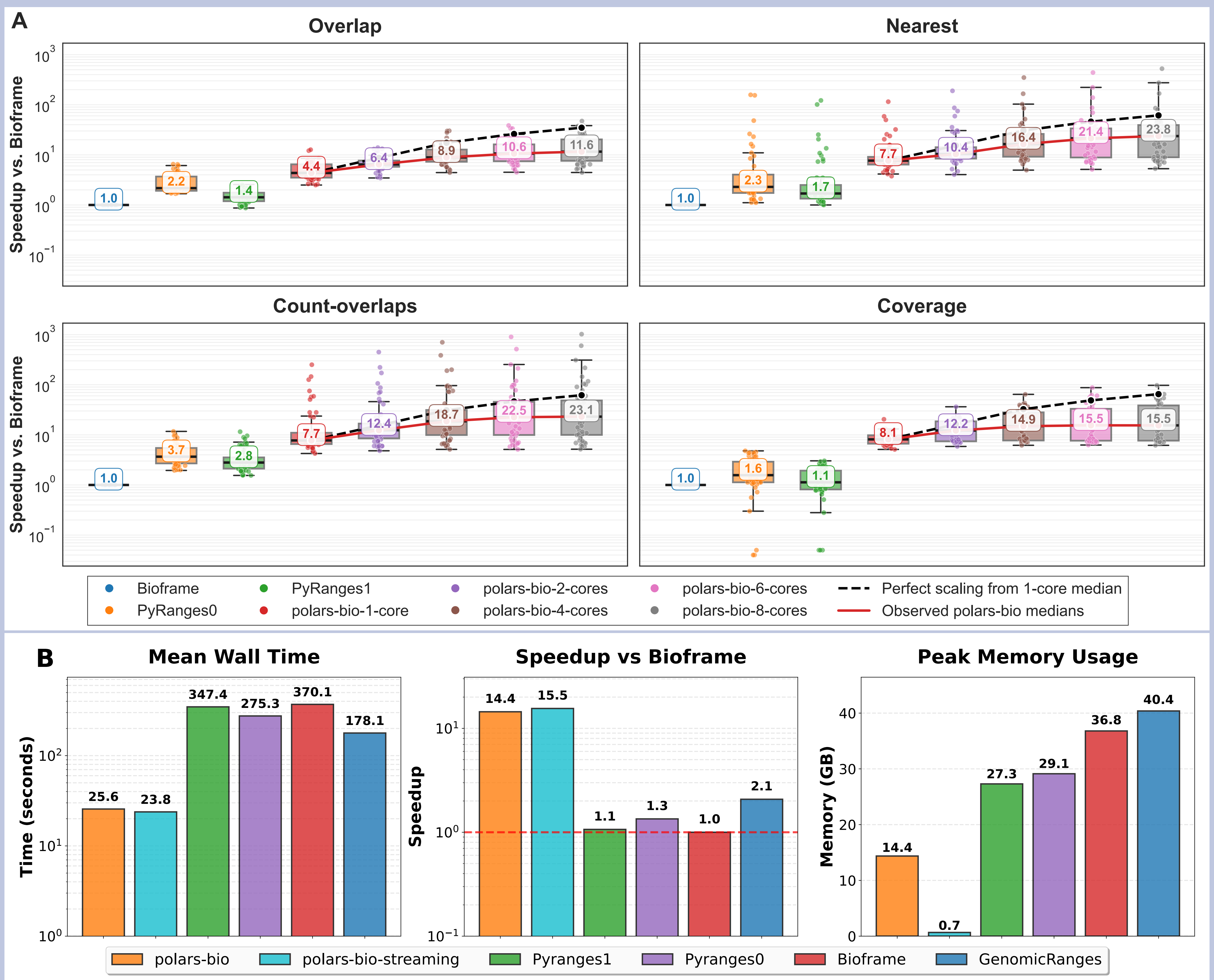
```
df_over=pb.overlap(
"gnomad_sv",
"v_gnomad")
```

```
df_cnt=pb.count_overlaps(
"v_local_bed",
"v_gnomad")
```

```
df_over.sink_csv(gs://target/overlap.csv)
```

Panel A—Main components of the *polars-bio* architecture and key technologies used.
Panel B—*polars-bio* integration points (Table Providers and Polars I/O plugin with **zero-copy** via Arrow C Stream FFI), predicate/projection pushdown in lazy plans, and an example dataflow demonstrating both DataFrame and SQL APIs.

Results



Panel A—Speedup distribution for genomic interval operations: 50 different pairs (combinations of datasets) from the AIList(4) benchmark; # overlaps: 5×10^4–10^9; inputs: 2×10^5–10^7 intervals.
Panel B—End-to-end overlap pipeline saving results in Parquet (# overlaps $\sim 3 \times 10^8$), *polars-bio* modes: (i) full materialization in memory prior to file export and (ii) streaming result batches.