Google Summer of Code 2025

CERN-HSF & Compiler Research Group

FINAL PRESENTATION

"Using ROOT in the field of genome sequencing"

Mentee Name:- Aditya Pandey

Mentors Name:- Vassil Vassilev, Martin Vassilev

THE CHALLENGE WITH GENOME DATA

- **Why ROOT for Genomics?**
- **Data Volume Crisis**: A single human genome at 30x coverage generates approximately 200 gigabytes of raw sequencing data, with current worldwide sequencing capacity exceeding 35 petabases per year.
- **Exponential Growth**: Genomics data acquisition is projected to reach more than one exabase of sequence per year within the next five years if growth continues at the current rate of doubling every seven months. By 2025, genomics is expected to be on par with or exceed astronomy, YouTube, and Twitter as one of the most demanding Big Data domain.
- **Performance Bottleneck**: Traditional formats (SAM/BAM/CRAM) struggle with random access queries on these massive datasets. A 30x whole genome BAM file consumes 80-90 gigabytes, and even modest studies with 1,000 samples require 80 terabytes of storage.
- **ROOT's Heritage**: CERN's ROOT framework has decades of experience processing petabytes of particle physics data daily, with proven columnar storage and compression technologies
- **Opportunity**: Apply ROOT's battle-tested high-energy physics data management expertise to solve genomics' data crisis, enabling faster queries and more efficient storage for population-scale studies

Our Solution

- **Applying ROOT's Data Framework to Genomics**
- **Core Approach**: RAMTools adapts ROOT's RNTuple columnar storage format - developed for particle physics data - to store and query genomic alignment data
- **Technical Solution**:
    - Convert SAM files from row-based to columnar format.
    - Store genomic fields (sequences, quality scores, positions) in separate columns.
    - Enable efficient compression and fast random access to specific regions.
- **Key Components**:
    - **samtoramntuple**: Conversion tool for SAM to RNTuple format.
    - **ramntupleview**: Query tool for extracting specific genomic regions.
    - **Chromosome splitting**: Option to split particular chromosome data.
- **Implementation**:
    - Built with C++
    - Command-line interface similar to samtools for easy adoption
    - Maintains full compatibility with standard SAM format fields
- **Design Goals**: Reduce storage requirements, improve query performance, and enable interactive analysis of large-scale genomic datasets

Technical Architecture

- **From SAM to RAM: The Conversion Pipeline**
- **Input Processing**:
  - Parse standard SAM/BAM files with custom streaming parser
  - Extract all mandatory fields and optional tags
  - Handle reference sequences and metadata
- **Columnar Transformation**:
  - Map SAM fields to RNTuple columns (qname, flag, position, sequence, quality)
  - Apply field-specific encoding: 2-bit packing for sequences, CIGAR compression
  - Store reference mappings and metadata separately
- **Compression Options**:
  - Multiple algorithms: ZSTD (default), LZ4 (fast queries), LZMA (best compression)
  - Quality score optimization: Full preservation, 8-bin quantization, or removal
  - Field-level compression tailored to data characteristics
- **Output Modes**:
  - Single file: All data in one ROOT file
  - Selective extraction: Filter specific chromosomes during conversion

Bottlenecks in Current Technology

Problem with Current formats

- **Storage bloat:** SAM is plain text; BAM/CRAM are still row-oriented with limited compression flexibility.
- **Slow region queries:** Require scanning or separate index file lookups; no sparse in-file index.
- **Row-based access:** Must materialize full records even when only a few fields are needed; poor cache locality.
- **Rigid compression:** Fixed strategies (BGZF, reference-based) and verbose quality scores; little configurability.

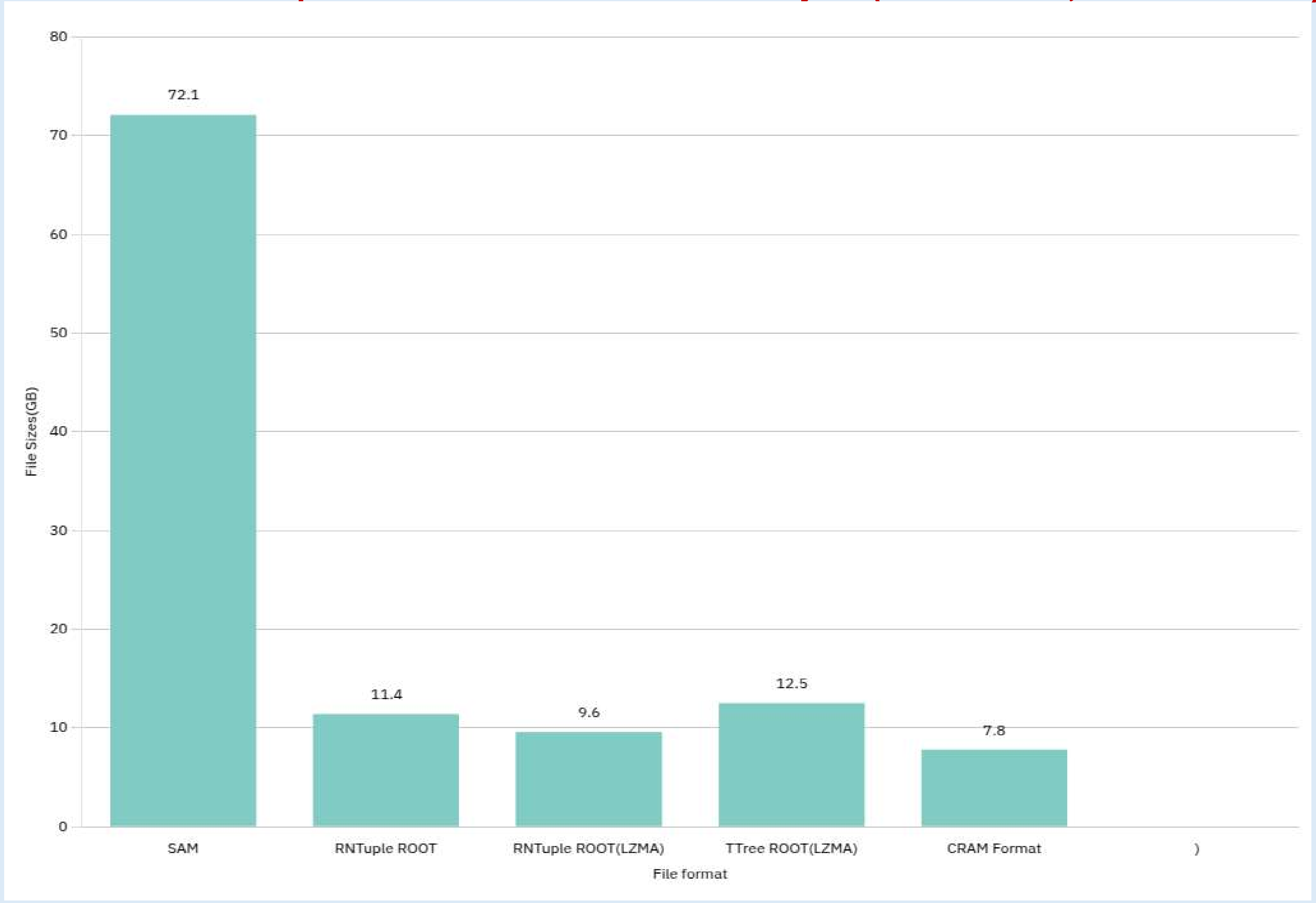
Our Goals

- **1. Create a Self-Contained Format** To eliminate the need for external reference files, simplifying data portability, sharing, and long-term archival.
- **2. Accelerate Query Performance** To drastically reduce analysis time, especially for the medium and large-scale queries common in modern genomic research.
- **3. Ensure High Compression** To achieve significant storage reduction from raw formats (like SAM) and remain competitive with other compressed formats.

Benchmark Results

File Size Comparison

HG00154 sample from the 1000 Genomes Project (196M reads, 72GB SAM file)



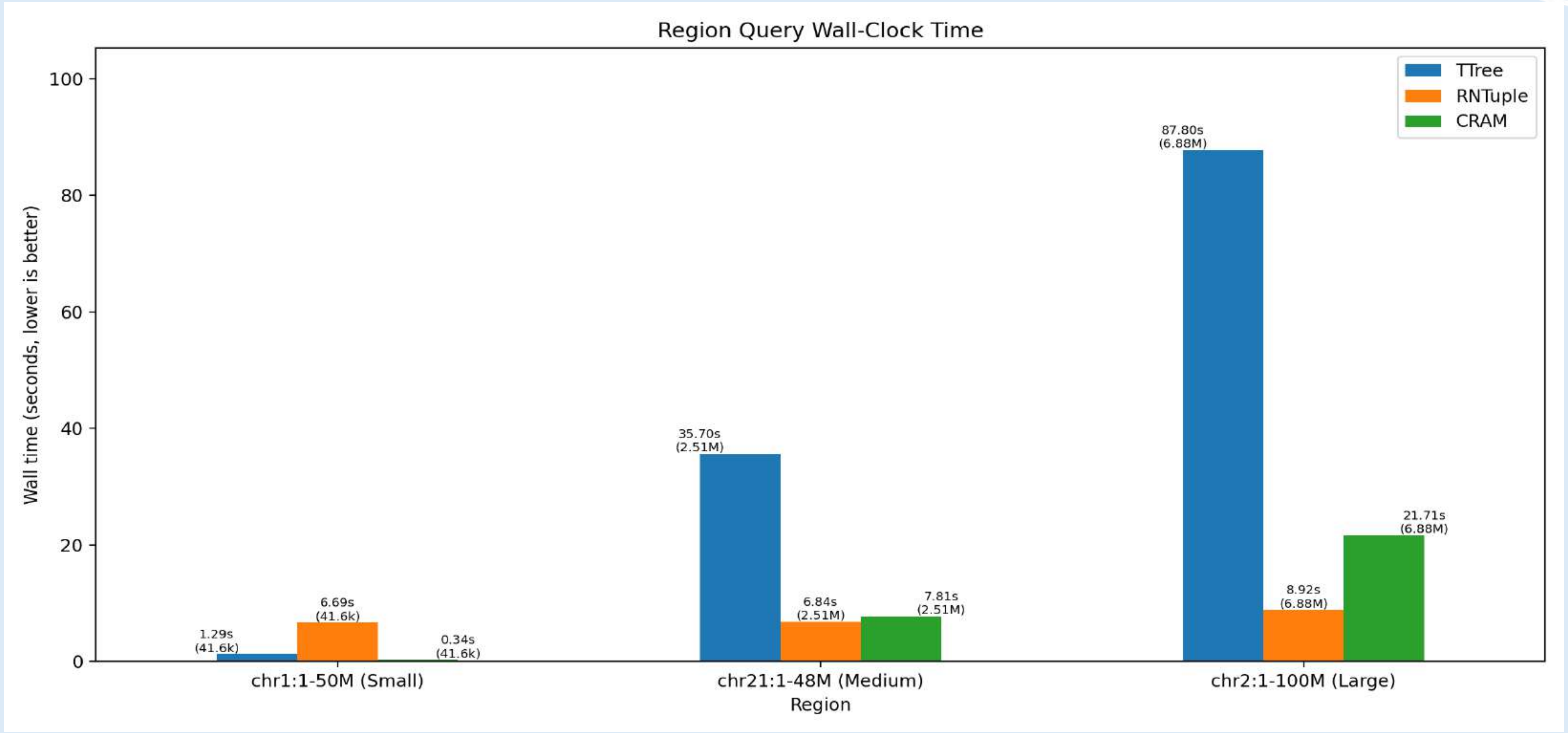
**CRAM format needs additional 3.2 GB reference file for querying.

**RNtuple ROOT has been converted using default ZSTD algorithm.

Benchmark Results

Region Querying Performance

HG00154 sample from the 1000 Genomes Project (196M reads, 72GB SAM file)



**CRAM format needs additional 3.2 GB reference file for querying.

Results Summary

Key Findings (HG00096 Dataset):

- **File Size (Compression from 72.1 GB SAM):**
  - **CRAM:** 7.8 GB (89% reduction) - Smallest file size.
  - **RNTuple (LZMA):** 10.8 GB (85% reduction) - Highly competitive compression.
- **Query Performance:**
  - **Small Region (41k reads):** CRAM is fastest (0.345s).
  - **Medium/Large Regions (2.5M+ reads):** RNTuple is significantly faster (e.g., 8.92s vs. 21.7s for CRAM on large regions).
- **The Critical Trade-Off: Portability**
  - **CRAM** achieves its small size by being reference-based. It **requires an external ~3.2 GB reference file** to be read, making its true cost (7.8 GB + 3.2 GB) and portability a key consideration.
  - **RNTuple** is fully **self-contained**. Its 10.8 GB file includes all data and requires no external dependencies.

Conclusion: RNTuple provides the best overall balance, offering strong compression and superior query performance for large-scale analysis, all within a single, portable file.

Thank you!