VCFC: Structural and Semantic Compression and Indexing of Genetic Variant Data

Kyle Ferriter*, Frank Mueller*, Amir Bahmani†, and Cuiping Pan‡
*North Carolina State University, Raleigh, NC, USA, mueller@cs.ncsu.edu
† Stanford Healthcare Innovation Lab, Stanford University, CA, USA, abahman@stanford.edu
‡ Palo Alto Epidemiology Research and Information Center for Genomics, VA Palo Alto, CA, USA, cuiping@stanford.edu

***Abstract*—Personalized genomic datasets are growing in size at an accelerating pace presenting a dilemma between the need for fast retrieval requiring "near data" and cost of storage, which decreases for "distant media" with larger capacity but longer access time. Instead of database technology, the bioinformatics community has developed an industry standard for compressing and indexing of genetic variant files that store the difference between a person's genome to a human reference genome. These standardizations rely on generic data compression schemes.**

This work contributes novel domain-specific compression and indexing algorithms that retain the structure and semantics of genetic variation data while supporting common query patterns. A line-based run-length partial compression technique for variant genotype data using a novel indexing strategy is developed and shown to perform well on large sample sets compared to the industry standard. The evaluation over genomic datasets indicates compression at a comparable size for our data representation while resulting in speedup of ≈2X in indexed queries compared to the industry standard. This underlines that our representation could replace existing standards resulting in reduced computational cost at equivalent storage size.

I. Introduction

In recent years, genomic datasets have rapidly grown in size as genetic sampling, sequencing, and analysis has become more integrated into common health and medical processes ranging from emergency room to periodic checkups [1], [2]. This places demands on data infrastructure for availability and short query responses while requiring large storage space. Hence, large medical data are being moved into the cloud, where storage scales and is widely available. Information of such data is accessed using indexing to select relevant records without reading in the entire data set.

This work specializes compression and indexing techniques of genetic data for genomic sequencing pipelines used in genetic variation research. This work makes the following contributions:

- A novel row-based run-length partial compression technique is developed with similar performance to block-based full compression.
- A row-based indexing strategy for genetic variant data is devised that leverages both key binning and sparse file offsets.
- Indexing using our novel data representation and algorithms is shown to result in 2X-3X superior performance over conventional storage techniques.

II. Background

Human DNA is largely assembled into 23 pairs of chromosomes, consisting of macromolecules represented symbolically by the letters A, T, C, and G. Genomic *sequencing pipelines* are used to identify the genetic makeup of an individual, resulting in DNA sequences recorded in variant calling format (VCF). Often, only DNA variations relative to the human reference genome and identified with sufficient accuracy/read depth [3] are recorded. Normally, these variations are in the range of 3-5 millions, in contrast to the 3 billion base pairs in the human reference genome. Fig. 1 depicts a reference chromosome (lower part) and the imaginary difference (colored) of an individual, where differences can be thought of as being tabulated (right side). These are stored in VCF files.

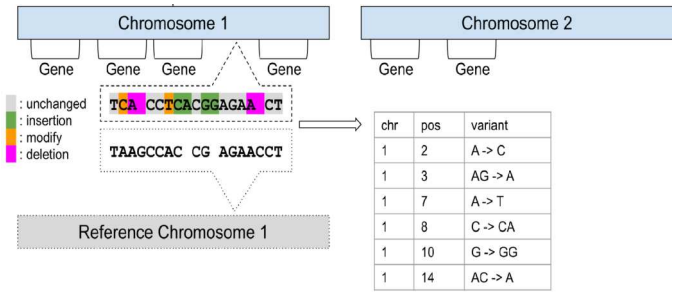

Fig. 1: Conceptual Description of Genome Differencing

VCF files are laid out as a mapping of genetic variation descriptions that may or may not contain variations, where an NxM matrix of N variants and M samples summarizes the differences. Fig. 2 depicts an excerpt of a VCF file with meta data followed by a sequence of homologous groups (HG) of individuals that have the alternate letter(s) in their primary or secondary strand when the encoding is non-zero (indicating the index of an alternate reference letter), otherwise (in case of zero) the sequence is identical to the reference.

III. Related Work

The most common **compression** format for VCF files is called `BGZF` and is part of the Samtools project [4]. This format aggregates VCF data lines and concatenates them in GZIP blocks, which means that any block needed by a query has to be decompressed before data can be processed.

9781-4244-3941-6/09/978-1-7281-6215-7/20/$31.00 © 2020 IEEE

#CHR	POS	ID	REF	ALT	...	HG01	HG03	HG04	HG05
1	10075	rs12	A	G	...	0|0	1|0	0|0	0|0
1	10115	rs21	G	A	...	0|0	0|0	1|1	0|0
...									
1	10838	rs44;rs46	GA	GAA,G	...	2|0	0|2	0|2	0|1

Fig. 2: Stylized Excerpt of a VCF File

Another common compression format is BCF, a binary format (documented page 27 of the VCFv3 specification [5]). It creates lookup keys in a metadata dictionary but otherwise concatenates variant description and sample genotyping before committing it to a compression block. Subsequent data processing often requires **indexing** into `BGZF` and `BCF` files, which is supported by the *Tabix* tool [6] within the Samtools suite. Tabix uses both a linear index and a six-level binning strategy, where each level from the top down uses logarithmically smaller bin sizes than the level above it. This significantly reduces seeks and limits the number of bytes read.

Other related work includes an assessment of domain-specific indexing for genomics data [7], which develops faster range-based lookups based on semantic information of genomic *regions* compared to generic/traditional techniques. Sparse indexing/binning has been used in databases [8], [9], where sparse indices are stored in dense files but not with filesystem support for sparse storage. QEMU [10] leverages sparse file support for reducing disk utilization for guest filesystems, yet without object keys or indexing. Nobre et al. [11] assess tensor unit acceleration capabilities of Nvidia Turing GPUs for binary neural networks calculating the Basian K2 score between SNPs to better understand associations between phenotypes and genotypes. Their work is complementary to ours as tensor unit acceleration can be equally applied to our VCFC representation, albeit without the overhead of prior decompression, again illustrating the advantage of our technique. Industry is developing APIs for native key-value SSD storage at the Flash Translation Layer (FTL) [12], which could be combined with our structural compression technique to become even more effective by offloading indexing into the FTL.

IV. Design

A. Compression (VCFC)

Our novel compression strategy, VCFC (VCF Compressed), focuses specifically on the M sample columns within a VCF file as the nine leading non-sample columns are (nearly) constant in size. VCFC takes advantage of underlying assumptions about the content of VCF files, namely:

1) There is more than 1 sample (patient) represented in the file. Typically, a few hundred up to many thousand samples are in a file with many identical values providing opportunities for run-length compression.
2) Alternate bases for a given variant are ordered in decreasing order of frequency.
3) Samples do not differ from the genome reference file in most positions and a genotype value of 0 is the most common case (same letter as the reference).

VCFC only compresses the sample columns as data size scales with more input data in the form of sample genomes, i.e., the N variant lines remain constant in the file while more columns M are added by new sample data. In particular, while the 1000 Genomes Project [13] has published nearly 90 million variants, M remains relatively small with a run-length of only up to 2504. If more samples are added, the size of the uncompressed dataset experiences a linear increase due to similarities of new samples to existing ones.

We determine that, within the 1000 Genomes Project chromosome 1 VCF file, genotype `0|0` is an order of magnitude more common than `0|1`, `1|0`, `1|1` followed by others at yet an order of magnitude lower rates. Hence, we restrict our strategy to only compress those 4 genotype values as they also provide the most significant opportunities for run-length compression as they tend to occur back-to-back.

Fig. 3 depicts, for the running example, the VCF file (top) with highlights indicating any potential for compression and our equivalent VCFC (bottom) with compressed data (one byte each consisting of flag and length bits). Here, the VCF file of 184 bytes was reduced to our VCFC file of 43 bytes. In larger VCF files, this ratio increases drastically as runs of repeated values become longer.

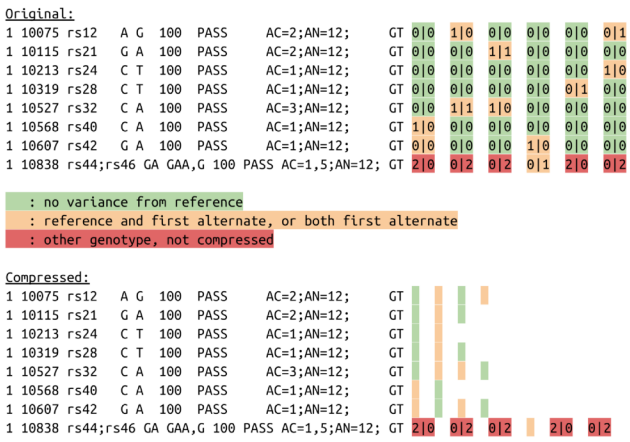

Fig. 3: VCFC Example Visual

B. Line-Based Contiguous Binned External Indexing

We have developed a novel data design for VCFC using compression based on an external, secondary index file. This external file stores a linear index of bins, each pointing to a record in the compressed VCFC file. The index entries map a chromosome and position to a byte offset within the VCFC compressed file of the line where that combination occurs. A pointer record encodes the chromosome, the position of the variant on the chromosome, and a byte offset. A binary search is employed for index queries since the entries in the index and the data in the file are both sorted.

V. Experimental Framework

A. Implementation Correctness

The evaluation of correctness for the VCFC compression and indexing techniques was performed via thorough experimentation. The first correctness was established by compressing a set of VCF files from the 1000 Genomes Project with VCFC, then decompressing with our VCFC, and checking the final output against the original VCF file with line-by-line differencing, which provided a perfect match. Further, a variety of VCF queries were issued via Tabix and it was validated that our technique matched the expected output.

B. Systems Evaluation

The VCFC implementation was realized in C++11, compiled using the GNU C++ compiler and libraries version `4.8.5` on CentOS 7 with Linux kernel `4.10.13`. For timing evaluations, optimization level 3 was enabled with `-O3`. No other nonstandard build flags were applied.

The evaluations were performed on a node of a cluster described in Tab. I. The implementation is single-threaded, i.e., only one core (irrespective of hyperthreading on/off) was used for execution. Only one execution was run at a time to minimize CPU context switching and cache contention and to eliminate I/O bandwidth contention.

Cluster Node 1:
• CPU: 2 × Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz, 16 cores, 32 threads (hyperthreading on). 20MiB L3 cache
• Storage: SAMSUNG MZPLK1T6HCHP-00003 (NVME), 1.6 TB

TABLE I: Cluster node used in evaluation experiments

Each timing-related evaluation was run 10 times and averaged. For tests with small number of data points, we report the standard deviation explicitly. For all other tests with hundreds data points or more, also averaged over 10 runs, the standard deviation for dense queries was observed to be a few hundred microseconds to occasionally a few milliseconds, and for sparse queries in the order of a few milliseconds.

C. Evaluation Data

The data used to evaluate the new compression and indexing strategies included both synthetic and real-world VCF objects.

Synthesized VCF files were created by generating N variants and M samples, with reference bases and alternate bases chosen at random, with up to A alternate bases with decreasingly probabilities, and with the same A for each variant in the VCF file. Synthetic files have shorter run lengths, which may underestimate the compression ratio compared to real data. Hence, synthetic data was only used for correctness evaluations.

Real-world VCF files were used both for evaluating correctness and for measuring storage and time metrics. The input files were from the 1000 Genomes Project [13] Phase 3 all-sample VCF files, which contain 2504 samples and approximately 89 million variants over all chromosomes.

VI. Results

Tab. II first depicts the compression ratio of the three schemes in the second column, which are all high and only a few percent apart. This underlines the effectiveness of our `VCFC`, which only compresses the sample columns for 0/1 combinations while variant site and description columns are not compressed. The latter can be queried without decompression, e.g., for filtering a chromosome or position range.

TABLE II: Compression ratio (in %), times (avg./std.dev. over 10 runs in sec.), VCF 1000 Genomes Project, chromosome 22

	ratio	time (seconds)
read	N/A	6.15 ± 0.24
BGZF	98.15%	127.29 ± 0.50
BCF	98.42%	238.43 ± 0.52
VCFC	96.87%	192.96 ± 0.82

The table also depicts the method-agnostic read overhead (row 2) of the uncompressed file, which is subsequently compressed in our binned external indexing format, which imposes compute overhead for the compression and write overhead for the respective device and filesystem.

Overheads are dominated by compression, where BCF compression took 1.9X as long as BGZF, and VCFC took 1.5X as long as BGZF. BCF performs two phases of compression, leading to more computational work. But VCFC has a much higher standard deviation than others since writes are on a line-by-line basis, making them smaller (latency oriented) and more numerous than the block-based writes (bandwidth oriented) of BGZF and BCF.

A. Indexing

We assess the cost of index queries for binned indices using different key phases. We perform a sensitivity study on binned indexing because the *bin size* is an input parameter for tuning compression resulting in a spectrum of overheads. We search through the index by seeking from the start of the bin in the compressed file and decompress lines only at the seek destination. For each query, we assess the time to search through the index to find the bin which contains a target position, P (depicted in blue in figures). Since we use a binary search algorithm, the number of bins searched increases logarithmic. Overall, the index search had relatively small overhead. Notice that the cost of reading meta overhead is omitted because the overhead is constant for every query.

Each bin size was evaluated with a uniform distribution of 200 position queries across the chromosome 22 VCF file, and each query was evaluated 10 times and averaged. In this first set of experiments, the kernel disk page cache was not flushed between runs as we wanted compression (computation) to dominate, not I/O (unlike in later experiments).

Fig. 5 shows the time (y-axis) of queries for different bin sizes (x-axis). Each bar reports the average over 2000 queries for index search (blue/bottom), decompression (orange/middle), and seek time (green/top). We observe that index search time contributes only a small constant cost to queries while compressing dominates the cost (2 ms) for small bin

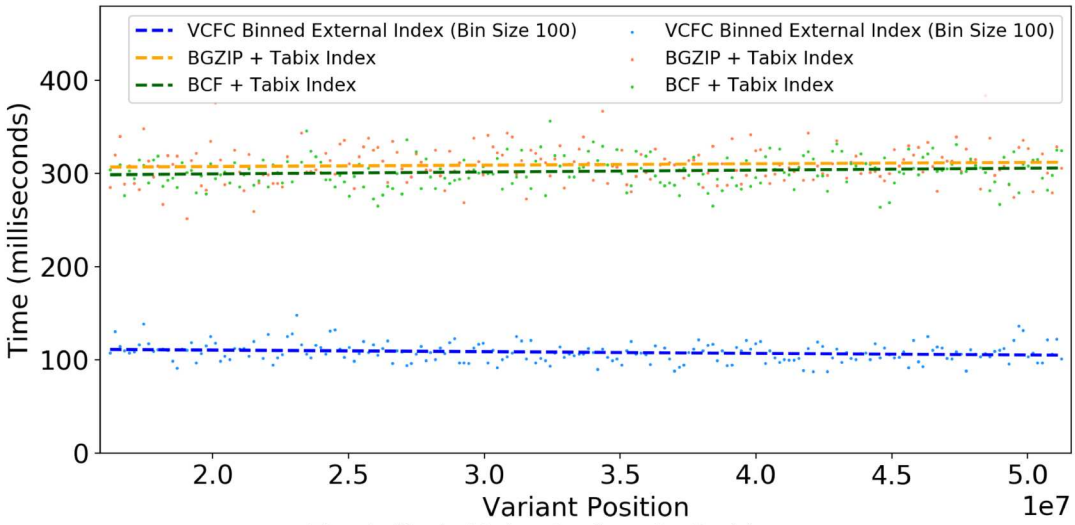

Fig. 4: Single Variant Lookups by Position

sizes while bin sizes of 450 and 1000 add a seek cost of another 1ms or 2ms, respectively. For reference, with the 1000 Genomes Project chromosome 22 VCF file the VCFC binned index with bin size 10 and 100 is 1.4 MiB and 140 KiB, respectively. An ideal bin size, e.g., 1000, is a trade-off between query performance and space consumption (as well as spatial locality if subsequent queries reference the same bin, as assessed in range-based queries below).

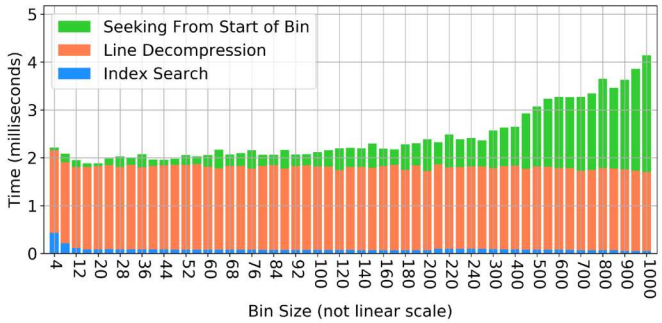

Fig. 5: Single variant query time profile related to index bin size (Node 1, NVME, EXT4)

B. Query Performance

Fig. 4 depicts the query performance (in ms, y-axis) for different variant positions (range $1.9 - 5.4 \times 10^7$, x-axis) and different techniques (see color-coded legend). For each data series, colored dots depicts single-query results approximated via (slightly increasing) dashed lines to reflect their near-range averages. The two reference techniques, BGZIP and BCF with Tabix indexing, result in query responses around 300ms while our VCFC approach averages just above 100ms, a 3X improvement as a result of our structured compression techniques. Variant positions have an insignificant impact of these times.

VII. Conclusion

This work showed that a novel encoding technique for genetic variant data based on structure- and semantic-aware compression results in comparable file sizes while novel indexing improves response times for common queries by 2X-3X. This underlines that our representation could replace existing standards resulting in reduced computational cost at equivalent storage size.

Acknowledgment

This work was supported in part by the NIH grant 1R01AR072013-01 and the Veterans Affairs Office of Research and Development Cooperative Studies Program. This research also received support by the generosity of Eric and Wendy Schmidt by recommendation of the Schmidt Futures program.

References

[1] R. Leach, "The Rise of Genomic Medicine," https://tedxgrandrapids.org/portfolio-item/the-rise-of-genomic-medicinerick-leach/, 2013.
[2] Stavropoulos DJ, Merico D, Jobling R, et al, "Whole genome sequencing expands diagnostic utility and improves clinical management in pediatric medicine," *NPJ Genomic Medicine*, 2016.
[3] Sims, D., Sudbery, I., Ilott, N. et al., "Sequencing depth and coverage: key considerations in genomic analyses," 2014.
[4] "Samtools," https://www.htslib.org/, Wellcome Trust Sanger Institute.
[5] "Vcf specification version 4.3," https://samtools.github.io/hts-specs/VCFv4.3.pdf, August 2019.
[6] H. Li, "Tabix: fast retrieval of sequence features from generic tab-delimited files," 2011.
[7] V. Jalili, M. Matteucci, J. Goecks, Y. Deldjoo, and S. Ceri, "Next generation indexing for genomic intervals," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 10, pp. 2008–2021, 2019.
[8] The PostgreSQL Global Development Group, "PostgreSQL BRIN Indexes," https://www.postgresql.org/docs/12/brin-intro.html, 2020.
[9] MongoDB, Inc, "Sparse Indexes," https://docs.mongodb.com/manual/core/index-sparse/, 2020.
[10] "QEMU disk image utility," https://www.qemu.org/docs/master/interop/qemu-img.html, Accessed on: 2020-04-26.
[11] R. Nobre, A. Ilic, S. Santander-Jiménez, and L. Sousa, "Exploring the binary precision capabilities of tensor cores for epistasis detection," in *International Parallel and Distributed Processing Symposium*, 2020, pp. 338–347.
[12] Yang Seok Ki, "Key Value SSD Explained – Concept, Device, System, and Standard," in *Storage Developer Conference*, 2017.
[13] The 1000 Genomes Project Consortium, "A global reference for human genetic variation," Nature 526, 68-74, 01 October 2015.