A Scalable Tool for Democratizing Variant Calling on Human Genomes Using Commodity Clusters

Khawar Shehzad
University of Missouri
Columbia, USA
khawar.shehzad@missouri.edu

Ajay Kumar
University of Missouri
Columbia, USA
ajay.kumar@missouri.edu

Matthew Schutz
University of Missouri
Columbia, USA
mjs2w6@missouri.edu

Chase Webb
University of Missouri
Columbia, USA
c.webb@missouri.edu

Polycarp Nalela
University of Missouri
Columbia, USA
polycarpnalela@missouri.edu

Manas Jyoti Das
Southern Illinois Univ. Edwardsville
Edwardsville, USA
madas@siue.edu

Praveen Rao
University of Missouri
Columbia, USA
praveen.rao@missouri.edu

Abstract

Variant calling is a fundamental task that involves identifying variants in an individual's genome compared to the reference genome. Knowing these variants is critical for assessing an individual's risk for diseases such as cancer and developing new treatments. Due to the large size of human genome sequences, processing and analyzing them requires significant compute and storage resources. Cluster computing is an attractive solution for processing a large workload of human genomes. In this paper, we present a scalable tool for *democratizing variant calling on human genome sequences* using testbeds that are available for academic research *at no charge*. Our tool can (a) execute two types of variant calling pipelines in a commodity cluster with CPUs and graphics processing units (GPUs); (b) enable improved cluster utilization and faster execution via asynchronous computations, minimal synchronization, and mutual exclusion when employing GPUs; and (c) execute variant calling pipelines of multiple users concurrently. Using publicly available human genome sequences, users can interactively experience the unique features of our tool, which has a low barrier to entry for large-scale variant calling.

CCS Concepts

• **Computing methodologies**; • **Applied computing**;

Keywords

Variant calling, human genomes, commodity clusters

CIKM '24, October 21–25, 2024, Boise, ID, USA

ACM ISBN 979-8-4007-0436-9/24/10
https://doi.org/10.1145/3627673.3679221

ACM Reference Format:
Khawar Shehzad, Ajay Kumar, Matthew Schutz, Chase Webb, Polycarp Nalela, Manas Jyoti Das, and Praveen Rao. 2024. A Scalable Tool for Democratizing Variant Calling on Human Genomes Using Commodity Clusters. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3627673.3679221

1 Introduction

With technological advances in whole genome sequencing (WGS) and lower sequencing cost, it is now feasible to employ WGS for clinical practice and large-scale genomic studies [14, 28]. This year the Sequence Read Archive (SRA) alone had 91 petabases of human genome data [24] indicating a rapid increase in the use of WGS in recent years. By analyzing an individual's genome, medical professionals can determine his/her risk for complex diseases (e.g., cancer) and develop effective treatment protocols.

Although the price of WGS has dropped over the years (e.g., $100 per genome [33]), the cost and efficiency of processing and analyzing human genome sequences has continued to pose new challenges [7]. This is because a whole genome sequence of an individual can consume gigabytes of storage space due to millions of reads. These reads are short (overlapping) fragments of the deoxyribonucleic acid (DNA) in the genome produced by a sequencer [18].

Variant calling is a fundamental task that involves identifying variants in an individual's genome compared to a reference genome [23]. There are different types of variants such as single nucleotide polymorphisms (SNPs), short insertions/deletions (indels), and structural variants [16]. For a single DNA sample, the pipeline involves several stages, namely, reading the large sequence data files, aligning the reads against a reference genome, additional pre-processing steps to mitigate sequencing errors, and executing a variant caller to produce raw variants [21]. When tumor and normal DNA samples are to be compared, the aligned reads for normal and tumor samples are analyzed to identify variants [21]. In general, the pipelines are compute and I/O intensive in nature.

There is continued interest in accelerating variant calling pipelines by leveraging parallel/distributed computing techniques and hardware accelerators. A few open source projects [20, 25, 26] have employed big data frameworks (e.g., Apache Hadoop [38], Apache Spark [42]) for variant calling on human genomes. Companies such as Google, NVIDIA, Illumina, and Sentieon are developing faster and more accurate solutions for human genome sequence analysis [17, 27, 37, 41]. In essence, there continues to be keen interest in accelerating human genome sequence analysis and reducing the overall processing cost. The availability of academic/experimental testbeds such as CloudLab [13] and FABRIC [4] provides a compelling opportunity for researchers and educators to conduct large-scale human genome analysis *at no charge*. To the best of our knowledge, none has explored this opportunity for genome analysis.

Motivated by the aforementioned reasons, we present a scalable tool that *democratizes variant calling on human genome sequences*. It is designed to enable efficient genome data processing for biomedical informatics applications. Users can execute two standard variant calling pipelines via our web-based tool *at no charge*. Our tool synergistically combines the heterogeneous resources on experimental testbeds while hiding their intricacies and complexity of setting up bioinformatics tools for users. Without loss of generality, our tool uses CloudLab and FABRIC testbeds that enable cluster computing. It supports concurrent requests from multiple users to execute the pipelines on different set of genome sequences. To deal with concurrent requests, our tool creates a single workload of genomes for the same variant calling pipeline to enable higher cluster utilization and more efficient execution of the pipeline.

At the heart of our tool, are two scalable techniques that we developed for accelerating variant calling pipelines using a commodity cluster on a large workload of genome sequences (i.e., AVAH [31, 32], AVAH* [10]). AVAH's novel design exploits asynchronous computations for executing different pipeline stages and has minimal synchronization leading to improved cluster utilization and faster execution of variant calling pipelines. On the other hand, AVAH* builds atop AVAH and is designed to effectively utilize a GPU-enabled cluster. It enables a pipeline stage to run on GPUs and/or CPUs based on their availability. It has a mutual exclusion scheme for correctly utilizing the GPUs in the cluster. AVAH and AVAH* enable good utilization of CPUs and GPUs in the cluster [10, 32]. New cluster resources can be dynamically added to our tool to scale with increasing number of users and their requests.

To the best of our knowledge, *no similar tool exists today* with the aforementioned capabilities providing a low barrier to entry for large-scale variant calling on human genomes–at no charge.

2 Background and Motivation

In this section, we present an overview of two standard variant calling pipelines and motivation for our tool.

Accelerating Variant Calling Using Cluster Computing/Hardware Accelerators. Early approaches used Apache Hadoop [1] and Apache Spark [2, 9, 43] to accelerate only the alignment stage. Others utilized field-programmable gate arrays (FPGAs) to accelerate alignment [3, 8]. Halvade [12] used MapReduce [11] to parallelize the variant calling pipeline of GATK, which is a widely adopted software for variant discovery. Later, GATK4 [20] was released that used Apache Spark for multithreading and parallelization for accelerating the variant calling pipeline. Nothaft et. al. [25, 26] created ADAM/Cannoli to handle large genomic datasets using Apache Spark/Apache Hadoop and parallelized the alignment process/variant calling by reusing existing tools. NVIDIA developed Parabricks to accelerate GATK pipelines using GPUs [27]. Google developed DeepVariant [41] that used deep learning for variant calling and operated directly on aligned reads. Recently, Illumina developed DRAGEN to accelerate the variant calling pipeline using FPGAs [37]. More recently, Sentieon developed highly optimized software-based algorithms for variant calling pipelines using CPUs and also a variant caller based on machine learning [17]. Recently, FPGAs were used to accelerate variant calling on viral genomes [40]. A more recent approach used smart network interface cards to enable secure variant calling on encrypted genome sequences/intermediate files to mitigate data breaches [30].

Germline and Somatic Variant Calling Pipelines. We consider two standard pipelines, namely, *germline* variant calling and *somatic* variant calling [21] for DNA sequencing. Figure 1(a) shows an example of a germline variant calling pipeline for a single DNA sample using GATK4. It involves four stages, namely, (i) reading files in the FASTQ format [39] containing raw unmapped reads and converting to the BAM format [35] containing unaligned reads, (ii) aligning reads with a reference genome [22] to produce mapped reads and marking duplicates, (iii) sorting the aligned reads and applying base quality score recalibration (BQSR) to correct sequencing errors, (iv) invoking HaplotypeCaller [20] to produce raw germline variants in the VCF format [36] (i.e., a text file).

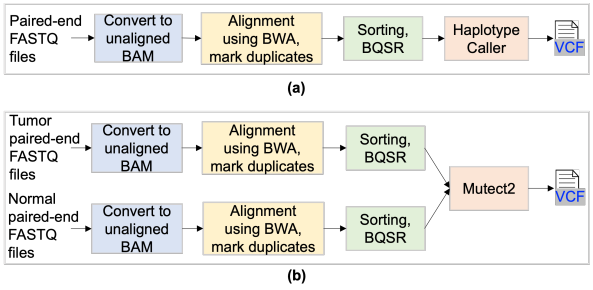


Figure 1: (a) Germline variant calling (b) Somatic variant calling

Figure1(b) shows an example of a somatic variant calling pipeline for tumor and normal samples. Tumor samples are expected to contain variants that could have caused the tumor. The FASTQ files of a tumor sample and normal sample are processed as before for the first three stages. Mutect2 [6] is finally invoked to produce the raw somatic variants.

Motivation. Prior work focused mainly on accelerating the variant calling pipeline on a single *gold-standard* high coverage human genome sequence [12, 20, 25, 27, 37, 41]. However, our work aims to accelerate the variant calling pipeline on *a large workload of human genome sequences* (e.g., provided by a hospital) using a commodity cluster. By improving the utilization of cluster resources (i.e., CPUs and GPUs) during the execution of the pipeline, we aim for faster execution of the pipeline on the input workload [10, 32]. Furthermore,

we aim to enable users to leverage the cluster computing resources available on academic testbeds [4, 13]–at no charge–rather than paying a high price to commercial cloud providers.

3 Our Tool

In this section, we describe our tool and its key components for democratizing variant calling on human genomes. Our tool builds atop our previous work designed to accelerate variant calling pipelines using commodity clusters, namely, AVAH [32] and AVAH★ [10]. While AVAH is designed to leverage only the CPUs of a cluster, AVAH★ is designed for a GPU-enabled cluster. Input genome sequences are read from the Hadoop Distributed File System (HDFS) of a cluster, and the output VCF files are written to HDFS. Figure 2 shows the overall architecture of our tool and its key components.

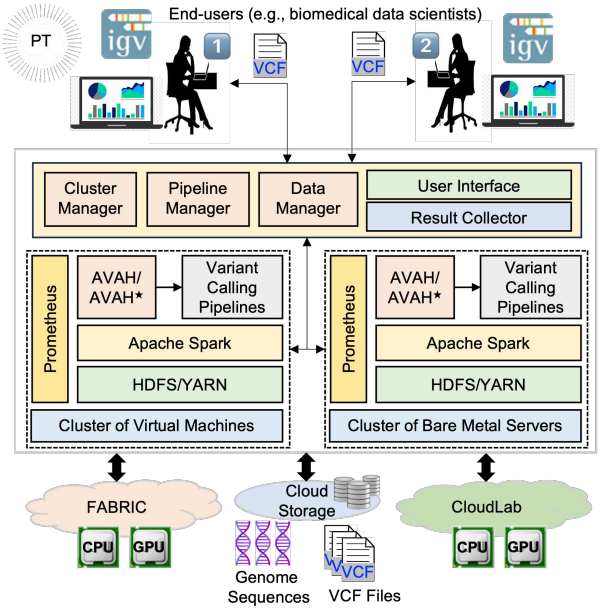


Figure 2: Overall architecture of our tool

AVAH. To overcome the poor cluster utilization of ADAM/Cannoli during variant calling [25], AVAH was proposed for efficient variant calling on a large workload of human genome sequences. AVAH improved the cluster utilization via the concept of futures, which enables non-blocking operations. It distributed the task of executing a variant calling pipeline on input sequences across the cluster nodes. It exploited task parallelism and data parallelism for different stages in the pipeline. In AVAH, each stage of a variant calling pipeline (on a sequence) is modeled as an atomic task. Tasks are executed as asynchronous computations using futures. Tasks representing the same pipeline stage managed by a Apache Spark executor (on a worker node of the cluster) are executed in a sliding window manner on small groups of sequences. This results in improved cluster utilization. (Each task/stage on a sequence is executed in a data parallel manner by re-using Spark-based APIs of ADAM/Cannoli or GATK4.) AVAH uses chaining of Spark's map operations on Spark's resilient distributed dataset (RDD) partitions (containing IDs of genome sequences) with only a single collect call at the end of the last stage of the pipeline introducing minimal synchronization. It was 3X-4.7X faster than ADAM/Cannoli for processing 98 low coverage human genome sequences using a 16-node cluster [32].

AVAH★. With availability of GPUs in today's cluster computing environments, AVAH★ was proposed for further accelerating variant calling using a GPU-enabled commodity cluster. AVAH★ builds atop AVAH's asynchronous computation model, map chaining, and minimal synchronization. However, it makes each asynchronous task GPU-aware and has two salient features: First, it enables a pipeline stage to execute either on a single cluster node's GPUs/CPUs or across multiple nodes using their CPUs. Second, it uses a mutual exclusion strategy for executing a pipeline stage of a sequence on the GPUs of a single node. As a result, the stages of other sequences in the same RDD partition can either wait for the GPUs to become available or proceed to use CPUs. Without mutual exclusion, pipeline stages would fail due to limited memory on the GPUs. AVAH★ also achieved high utilization of the cluster CPUs and GPUs. On a 8-node cluster (with a total of 8 GPUs), AVAH★ was 3.6X-5X faster than AVAH, which used only CPUs, for processing the aforementioned sequences.

For germline variant calling, each stage for a DNA sample (shown in Figure 1(a)) is modeled and executed by one asynchronous task in AVAH/AVAH★. However, for somatic variant calling, each stage for a tumor and its normal sample (shown in Figure 1(b)) is together modeled and executed by one asynchronous task in AVAH/AVAH★.

Cluster Manager (CM). CM tracks the different clusters allocated on FABRIC [4] and CloudLab [13] for executing variant calling. On FABRIC, a cluster is comprised of virtual machines (VMs); however, on CloudLab, a cluster has baremetal servers. A cluster may or may not have any GPUs based on available resources. Note that the availability of GPUs is higher on FABRIC compared to CloudLab. New clusters can be added dynamically when required. Hence, our tool can scale with increasing demand from users.

Pipeline Manager (PM). PM is responsible for launching the appropriate pipeline (i.e., germline, somatic) on a selected cluster. If the user does not specify the cluster, PM selects an underutilized cluster that has the least number of sequences queued for processing. PM runs AVAH/AVAH★ continuously on each cluster. A set of genome sequences from different users are batched and processed. After completion of the current batch, the next batch is processed. This is because AVAH/AVAH★ are designed to achieve high cluster utilization on a workload of genome sequences rather than a single sequence at a time.

Data Manager (DM). DM can retrieve sequences from cloud storage and copy them to HDFS of a cluster. It can also download sequences from SRA [24] and the European Nucleotide Archive [15]. Cloud storage is also used to store the output VCF files of AVAH/AVAH★ so that a user can download his/her files, view them, and generate a phylogenetic tree (PT) [29] to understand the relationship between different sequences.

Cluster Usage Metrics Collection/Visualization. Prometheus[1], an open source monitoring system, is used to collect metrics related to

[1] https://prometheus.io/

CPU/GPU/memory usage, disk I/O, and network throughput. The metrics are visualized using Grafana[2]. Hence, a user can continuously observe the load on different clusters before choosing the one for variant calling.

4 Demonstration Scenarios

Both AVAH and AVAH* were implemented in Scala (2.12.8) using Apache Spark (2.4.7) and Apache Hadoop (2.7.6). GATK4 (4.1.8.0) was used for CPU-based implementation of the variant calling pipelines. NVIDIA Parabricks (4.0.0) was used for the GPU-based implementation of GATK4. The user interface (UI) was implemented using Django (4.2.2), JavaScript, CSS, and HTML. Prometheus (2.47) and Grafana (10.1) were used for real-time monitoring of the cluster resources. For viewing the VCF files in the UI, IGV's JavaScript implementation [34] was used. For PT construction, we used Google Drive as a shared cloud storage for input genome sequences and VCF files produced by the variant calling pipelines.

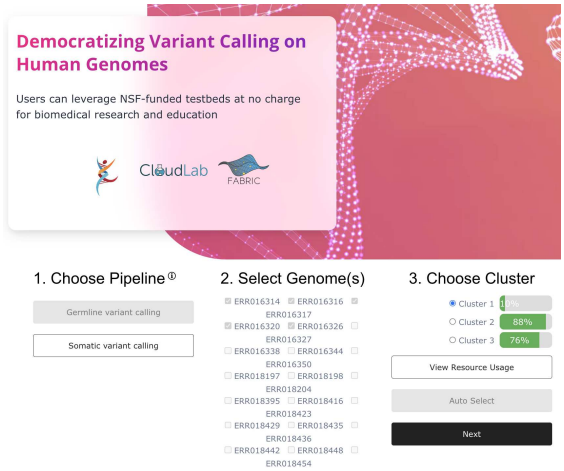
The following four scenarios will be demonstrated using publicly available *de-identified* human genome sequences. (Hence, data security is not a major concern for the demo.)

Scenario 1. In this scenario, we will demonstrate how a user can execute variant calling pipelines using our tool. The user will first select germline variant calling. After this, the user will select the input genome sequences to process from the 1000 Genomes Project [19]. (A list of accession IDs of sequences can also be provided by the user.) The user can then select a specific cluster after observing the load of the different clusters and their hardware configurations. For example, some clusters may have GPUs for faster execution. Finally, the variant calling pipeline is submitted to the selected cluster. A screenshot is shown in Figure 3(a). The UI is updated so that the user can observe the stages that have completed for a sequence. The user is notified via the UI once the output VCF files are available on cloud storage. The VCF files can be downloaded or viewed using IGV. Furthermore, a PT can be constructed on the VCF files. Some partial screenshots are shown in Figure 3(b).

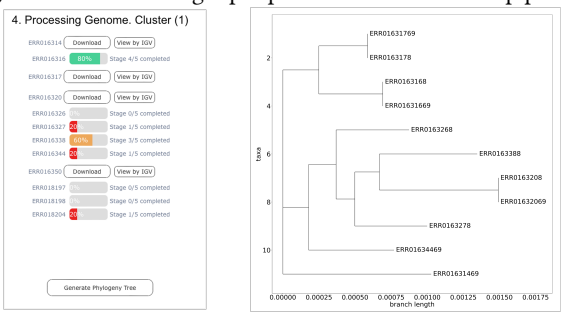
Scenario 2. In this scenario, we will demonstrate how our tool handles concurrent requests from multiple users. Multiple requests will be submitted concurrently to execute the same germline variant calling pipeline (but on different sequences) on the same cluster. These requests will be queued in the selected cluster. During the next round of execution of AVAH (or AVAH*), the queued sequences will be executed together as a batch as AVAH (or AVAH*) is designed to maximize cluster utilization on a workload of sequences. Each user will be notified as soon as the desired output VCF files are available on cloud storage. Our goal is to reduce the total wait time for the users while maximizing the usage of cluster resources.

Scenario 3. In this scenario, we will demonstrate our tool's ability to automatically select a cluster for the user. The user can select the germline variant calling pipeline and genome sequences. Our tool will observe the load on the different clusters (based on metrics collected by Prometheus) and the number of sequences queued for processing on each cluster. It will automatically select an appropriate cluster for the user's request to maximize the overall utilization of the testbeds' resources. This feature is useful when the user has no preference on the hardware configurations allowing our tool to optimize the execution of concurrent requests.

[2] https://grafana.com/



(a) The UI for selecting input parameters to execute a pipeline



(b) The UI showing processing stages and the constructed PT

Figure 3: Screenshots of our tool

Scenario 4. In this scenario, we will demonstrate our tool's ability to perform somatic variant calling. The user can select from a list of publicly available tumor/normal samples from the Texas Cancer Research Biobank [5] and the chromosome of interest (e.g., Chromosome 1). After choosing a cluster, the selected samples will be processed using AVAH/AVAH* according to the pipeline shown in Figure 1(b). The user will be notified via the UI once the output VCF files are available on cloud storage.

5 Conclusion

We presented a tool for democratizing variant calling on human genomes using commodity clusters in two experimental testbeds. It provides a low barrier to entry for large-scale variant calling on human genome sequence using testbeds available for academic research at no charge. The project code is available at https://github.com/MU-Data-Science/GAF.

Acknowledgments. This work was supported by the National Science Foundation under Grant Nos. 2201583 and 2034247.

References

[1] José M Abuín, Juan C Pichel, Tomás F Pena, and Jorge Amigo. 2015. BigBWA: Approaching the Burrows-Wheeler Aligner to Big Data Technologies. *Bioinformatics* 31, 24 (2015), 4003–4005.

[2] José M Abuín, Juan C Pichel, Tomás F Pena, and Jorge Amigo. 2016. SparkBWA: Speeding up the Alignment of High-Throughput DNA Sequencing Data. *PLoS ONE* 11, 5 (2016).

[3] Nauman Ahmed, Vlad-Mihai Sima, Ernst Houtgast, Koen Bertels, and Zaid Al-Ars. 2015. Heterogeneous Hardware/Software Acceleration of the BWA-MEM DNA Alignment Algorithm. In *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 240–246.

[4] Ilya Baldin, Anita Nikolich, James Griffioen, Indermohan Inder S. Monga, Kuang-Ching Wang, Tom Lehman, and Paul Ruth. 2019. FABRIC: A National-Scale Programmable Experimental Network Infrastructure. *IEEE Internet Computing* 23, 6 (2019), 38–47.

[5] Lauren B Becnel, Stacey Pereira, Jennifer A Drummond, Marie-Claude Gingras, Kyle R Covington, Christie L Kovar, Harsha Vardhan Doddapaneni, Jianhong Hu, Donna Muzny, Amy L McGuire, et al. 2016. An Open Access Pilot freely Sharing Cancer Genomic Data From Participants in Texas. *Scientific Data* 3, 1 (2016), 1–10.

[6] David Benjamin, Takuto Sato, Kristian Cibulskis, Gad Getz, Chip Stewart, and Lee Lichtenstein. 2019. Calling Somatic SNVs and Indels with Mutect2. *BioRxiv* (2019), 861054.

[7] Bonnie Berger and Yun William Yu. 2023. Navigating Bottlenecks and Trade-offs in Genomic Data Analysis. *Nature Reviews Genetics* 24, 4 (2023), 235–250.

[8] Yu-Ting Chen, Jason Cong, Zhenman Fang, Jie Lei, and Peng Wei. 2016. When Apache Spark Meets FPGAs: A Case Study for Next-Generation DNA Sequencing Acceleration. In *Proc. of the 8th USENIX Conference on Hot Topics in Cloud Computing* (Denver, CO). 64–70.

[9] Jason Cong, Jie Lei, Sen Li, Myron Peto, P. Spellman, Peng Wei, and Peipei Zhou. 2015. CS-BWAMEM: A Fast and Scalable Read Aligner at the Cloud Scale for Whole Genome Sequencing. In *High Throughput Sequencing Algorithms and Applications (HITSEQ)*.

[10] Manas Das, Khawar Shehzad, and Praveen Rao. 2023. Efficient Variant Calling on Human Genome Sequences Using a GPU-Enabled Commodity Cluster. In *Proc. of 32nd ACM International Conference on Information and Knowledge Management (CIKM)*. 3843–3848.

[11] Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified Data Processing on Large Clusters. In *Proc. of the 6th OSDI Conference*. 137–150.

[12] D. Decap, J. Reumers, C. Herzeel, P. Costanza, and J. Fostier. 2015. Halvade: Scalable Sequence Analysis with MapReduce. *Bioinformatics* 31, 15 (2015), 2482–2488.

[13] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, Aditya Akella, Kuangching Wang, Glenn Ricart, Larry Landweber, Chip Elliott, Michael Zink, Emmanuel Cecchet, Snigdhaswin Kar, and Prabodh Mishra. 2019. The Design and Operation of CloudLab. In *2019 USENIX Annual Technical Conference* (Renton, WA). 1–14.

[14] COVID Human Genetic Effort. 2020. The COVID Human Genetic Effort: Our Mission. https://www.covidhge.com/.

[15] ENA. 2024. European Nucleotide Archive. https://www.ebi.ac.uk/ena.

[16] Ensembl. 2021. Variant Classification. https://m.ensembl.org/info/genome/variation/prediction/classification.html.

[17] Donald Freed, Renke Pan, Haodong Chen, Zhipan Li, Jinnan Hu, and Rafael Aldana. 2022. DNAscope: High Accuracy Small Variant Calling Using Machine Learning. *bioRxiv 2022.05.20.492556* (2022).

[18] Sara Goodwin, John D McPherson, and W Richard McCombie. 2016. Coming of Age: Ten Years of Next-Generation Sequencing Technologies. *Nature Reviews Genetics* 17, 6 (2016), 333–351.

[19] IGSR. 2015. 1000 Genomes Phase 3 Release. https://www.internationalgenome.org/data-portal/data-collection/phase-3.

[20] Broad Institute. 2023. GATK4. https://github.com/broadinstitute/gatk.

[21] Daniel C. Koboldt. 2020. Best Practices for Variant Calling in Clinical Sequencing. *Genome Medicine* 12, 1 (2020), 91.

[22] Heng Li. 2013. Aligning Sequence Reads, Clone Sequences and Assembly Contigs With BWA-MEM. *arXiv preprint arXiv:1303.3997* (March 2013).

[23] NCBI. 2013. Genome Reference Consortium Human Build 38. https://www.ncbi.nlm.nih.gov/datasets/genome.

[24] NCBI. 2024. Sequence Read Archive. https://www.ncbi.nlm.nih.gov/sra/.

[25] Frank A. Nothaft. 2017. *Scalable Systems and Algorithms for Genomic Variant Analysis*. Ph. D. Dissertation. UC Berkeley, ProQuest.

[26] Frank Austin Nothaft, Matt Massie, Timothy Danford, Zhao Zhang, Uri Laserson, Carl Yeksigian, Jey Kottalam, Arun Ahuja, Jeff Hammerbacher, Michael D. Linderman, Michael J. Franklin, Anthony D. Joseph, and David A. Patterson. 2015. Rethinking Data-Intensive Science Using Scalable Analytics Systems. In *Proc. of the 2015 ACM SIGMOD Conference* (Victoria, Australia). 631–646.

[27] Kyle A. O'Connell, Zelaikha B. Yosufzai, Ross A. Campbell, Collin J. Lobb, Haley T. Engelken, Laura M. Gorrell, Thad B. Carlson, Josh J. Catana, Dina Mikdadi, Vivien R. Bonazzi, and Juergen A. Klenk. 2023. Accelerating Genomic Workflows Using NVIDIA Parabricks. *BMC Bioinformatics* 24 (2023).

[28] All of Us Research Program Investigators. 2019. The "All of Us" Research Program. *New England Journal of Medicine* 381, 7 (2019), 668–676.

[29] Edgardo Ortiz. 2019. vcf2phylip v2.0: Convert a VCF Matrix Into Several Matrix Formats for Phylogenetic Analysis. https://github.com/edgardomortiz/vcf2phylip.

[30] Praveen Rao and Khawar Shehzad. 2024. A Technique for Secure Variant Calling on Human Genome Sequences Using SmartNICs. In *Proc. of the 17th IEEE International Conference on Cloud Computing (CLOUD 2024)*. 1–8.

[31] Praveen Rao and Arun Zachariah. 2022. Enabling Large-Scale Human Genome Sequence Analysis on CloudLab. In *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 1–2.

[32] Praveen Rao, Arun Zachariah, Deepthi Rao, Peter Tonellato, Wesley Warren, and Eduardo Simoes. 2021. Accelerating Variant Calling on Human Genomes Using a Commodity Cluster. In *Proc. of 30th ACM International Conference on Information and Knowledge Management (CIKM)*. 3388–3392.

[33] Antonio Regalado. 2020. China's BGI Says It Can Sequence a Genome for Just $100. *MIT Technology Review, February* 26 (2020), 2020.

[34] James T Robinson, Helga Thorvaldsdottir, Douglass Turner, and Jill P Mesirov. 2022. igv.js: An Embeddable JavaScript Implementation of the Integrative Genomics Viewer (IGV). *Bioinformatics* 39, 1 (12 2022), btac830.

[35] Samtools. 2021. Sequence Alignment/Map Format Specification. https://samtools.github.io/hts-specs/SAMv1.pdf.

[36] Samtools. 2021. The Variant Call Format (VCF) Version 4.2 Specification. https://samtools.github.io/hts-specs/VCFv4.2.pdf.

[37] Konrad Scheffler, Severine Catreux, Taylor O'Connell, Heejoon Jo, Varun Jain, Theo Heyns, Jeffrey Yuan, Lisa Murray, James Han, and Rami Mehio. 2023. Somatic Small-Variant Calling Methods in Illumina DRAGEN™ Secondary Analysis. *bioRxiv 2023.03.23.534011* (2023).

[38] Tom White. 2009. *Hadoop: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

[39] Wikipedia. 2000. FASTQ Format. https://en.wikipedia.org/wiki/FASTQ_format.

[40] Tiancheng Xu, Scott Rixner, and Alan L. Cox. 2023. An FPGA Accelerator for Genome Variant Calling. *ACM Transactions on Reconfigurable Technology and Systems* (May 2023), 1–20.

[41] Taedong Yun, Helen Li, Pi-Chuan Chang, Michael F Lin, Andrew Carroll, and Cory Y McLean. 2021. Accurate, Scalable Cohort Variant Calls Using DeepVariant and GLnexus. *Bioinformatics* 36, 24 (2021), 5582–5589.

[42] Matei Zaharia, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. 2010. Spark: Cluster Computing with Working Sets. In *Proc. of the 2nd USENIX Conference on Hot Topics in Cloud Computing*. Boston.

[43] Lingqi Zhang, Cheng Liu, and Shoubin Dong. 2019. PipeMEM: A Framework to Speed Up BWA-MEM in Spark with Low Overhead. *Genes* 10, 11 (2019).