Invited: Accelerating Genome Analysis via Algorithm-Architecture Co-Design

Onur Mutlu Can Firtina
ETH Zürich

High-throughput sequencing (HTS) technologies have revolutionized the field of genomics, enabling rapid and cost-effective genome analysis for various applications. However, the increasing volume of genomic data generated by HTS technologies presents significant challenges for computational techniques to effectively analyze genomes. To address these challenges, several algorithm-architecture co-design works have been proposed, targeting different steps of the genome analysis pipeline. These works explore emerging technologies to provide fast, accurate, and low-power genome analysis.

This paper provides a brief review of the recent advancements in accelerating genome analysis, covering the opportunities and challenges associated with the acceleration of the key steps of the genome analysis pipeline. Our analysis highlights the importance of integrating multiple steps of genome analysis using suitable architectures to unlock significant performance improvements and reduce data movement and energy consumption. We conclude by emphasizing the need for novel strategies and techniques to address the growing demands of genomic data generation and analysis.

1. Introduction

Genome analysis plays a crucial role in various fields such as personalized medicine [1], agriculture [2], evolutionary biology [3], pharmacogenomics [4], infectious disease control [5, 6], cancer research [7] and microbiome studies [8]. The advent of high-throughput sequencing (HTS) technologies, such as sequencing-by-synthesis (SBS) [9], Single Molecule Real-Time (SMRT) [10], and nanopore sequencing [11–13], has revolutionized genome analysis, enabling faster and more cost-effective sequencing of genomes by generating a large amount of genomic data at relatively low cost [14]. However, the analysis of genomic data is challenging due to a variety of reasons: 1) HTS technologies can only sequence relatively short fragments of genomes, called *reads*, whose locations in the entire genome are unknown, 2) these reads can contain *sequencing errors* [14, 15], leading to differences from their original sequences, 3) the sequenced genome may not (and usually does not) exactly match recorded genomes in a reference database, known as *reference genomes*, due to variations between individuals within and across species. Despite significant improvements in computational tools since the 1980s [16] to overcome such challenges, the rapid growth in genomic data [17] has led to ever larger computational overheads in the genome analysis pipeline, posing large challenges for efficient and timely analysis of genomes [18, 19].

A genome analysis pipeline consists of multiple key steps, each of which affects the accuracy, speed, and energy consumption of genome analysis. First, *basecalling* translates the *raw sequencing data* that HTS generates (e.g., measured electrical signals in nanopore sequencing) into sequences of genomic characters (e.g., A, C, G, and Ts in DNA). Basecalling is time-consuming because it relies heavily on compute-intensive approaches that process large chunks of noisy and error-prone raw data to accurately infer the actual nucleotide sequences [13, 19–24]. Second, *real-time analysis of raw sequencing data* [5, 25–34] aims to analyze the reads simultaneously while the read is being sequenced using a particular sequencing technology (e.g., nanopore sequencing). Although real-time analysis of raw sequencing data provides enormous advantages in significantly reducing the overall genome analysis time and cost [25], it introduces unique challenges as the analysis needs to match stringent throughput and latency requirements to satisfy *real-time* requirements [34]. Third, *read mapping* aims to find similarities and differences between genomic sequences (e.g., between sequenced reads and reference genomes of one or more species). Read mapping includes several steps such as sketching [35–40], seeding [41–49], and alignment [50–55], which demand considerable processing power and memory due to the large scale of genomic sequences [16, 56, 57]. Fourth, subsequent steps of the genome analysis (i.e., *downstream analysis*) use the output generated in the read mapping step. An example of such downstream analysis is known as *variant calling* [58–64], which aims to identify genetic differences, known as *variants*, between an individual's genome and a reference genome. Variant calling is often followed by additional steps, such as *gene annotation* [65–69] and *enrichment analysis* [70–73]. These steps aim to generate insights from the identified variants and determine if these variants show an unexpectedly high or low statistical correlation with specific functional behavior (e.g., association with a disease) that can be used in a clinical report [74].

Many pure algorithmic and software techniques aim to address the computational challenges in the genome analysis pipeline. These works improve the performance and accuracy of the computational tools by 1) reducing overall computational and space complexity [55, 75], 2) eliminating useless work [38, 43–45, 56, 57, 76–78], 3) optimizing data structures and memory access patterns [79–81], 4) exploiting parallelism in multi-core, many-core, and SIMD architectures [38, 44, 77, 78, 82–86], and 5) employing machine learning techniques [15, 64, 77, 78]. These works fall short on greatly improving performance and energy consumption due to at least three major reasons. First, many of these approaches incur significant data movement between computation units and memory units [18, 87]. Second, a large portion of the data becomes useless in downstream genome analysis [88], and performing computation on it wastes time and energy. Third, HTS technologies produce sequencing data at an increasingly high rate, which makes it challenging to keep up with the throughput of these sequencing technologies, especially in time-critical scenarios [18, 34].

Since software techniques alone are not effective enough at coping with huge amounts of genomic data and the stringent requirements of genome analysis, it is critical to design software-hardware cooperative techniques to accelerate genome analysis. To this end, several works co-design algorithms and architectures to substantially improve the performance and energy efficiency of the genome analysis pipeline. These works 1) reduce data movement overheads by employing processing in memory (PIM) [2, 89–106], or processing near storage (e.g., solid-state drives) [87] and 2) efficiently co-design and execute computationally complex algorithms with massive parallelism and efficient hardware design using specialized architectures, e.g., field programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs) [31, 46, 48, 49, 54, 84, 107–123]

In this paper (and the associated invited talk), we review the recent advancements in accelerating genome analysis via algorithm-architecture co-design and discuss emerging challenges that highlight the need for new acceleration techniques. We aim to provide a brief yet comprehensive overview of the current state of the field and inspire future research directions to further improve the efficiency of genome analysis and hopefully enable new use cases and computing platforms.

2. Accelerating Basecalling

HTS technologies produce raw sequencing data, the content of which depends on the type of sequencing technology employed. There are three main types of sequencing technologies: sequencing by synthesis (SBS) [9], Single Molecule Real-Time (SMRT) [10], and nanopore sequencing [11]. SBS generates images where the color intensity at a particular position of an image represents

the base of the read. Basecalling after SBS aims to accurately associate these colors with their corresponding bases while correcting sequencing errors [124]. SMRT sequencing generates continuous images in a movie format by sequencing the same read multiple times via a strategy known as circular consensus sequencing (CCS) [125]. Although these images can be quickly converted to their corresponding bases, the high noise associated with SMRT sequencing requires additional steps to correct sequencing errors [125]. These techniques include alignment [47], consensus assembly construction [125], and polishing [15, 126]. Nanopore sequencing generates raw electrical signals as DNA or RNA molecules pass through tiny pores (i.e., nanoscale holes) called *nanopores* [11]. Changes in ionic current, measured as nucleotides pass through, are sampled in real-time and used to perform 1) basecalling and 2) real-time genome analysis.

Recent basecalling works [22, 24, 77, 78, 127–132] especially focus on basecalling raw nanopore signals due to two major reasons. First, the measured signal represents a combination of *multiple nucleotides* passing through the nanopore, making the basecalling task more challenging compared to the relatively simpler and more direct signal-to-base conversion in SBS and SMRT sequencing methods [19, 78]. Second, nanopore sequencing provides unique opportunities for real-time genome analysis that can be used to reduce the time and cost of sequence analysis [19, 34], as we discuss in §3.

Basecalling techniques developed for nanopore sequencing mainly use deep neural networks (DNNs) [78] to achieve high accuracy. However, these methods are computationally expensive to train and use with large amounts of raw electrical signal data [88]. To address this issue, several algorithm-architecture co-design works have been proposed. First, some works accelerate the execution of DNN operations using graphics processing units (GPUs) [22, 24, 127–132]. GPUs can substantially improve basecaller performance by providing massive parallelism for performing matrix multiplications in DNNs. Second, RUBICON [78] and TargetCall [77] reduce unnecessary computations in GPU-based basecallers by 1) reducing the DNN parameters and precision [78] or 2) introducing pre-basecalling filters [77]. Third, several works use processing-in-memory (PIM) [88, 96, 133], or FPGAs [119] to accelerate basecalling and reduce power consumption. A recent work that uses PIM, GenPIP [88], shows that a significant portion of useless data can propagate to downstream analysis, causing unnecessary data movement, compute cycles, and energy consumption. To eliminate such useless operations, GenPIP combines *both* basecalling and read mapping in PIM to quickly identify unnecessary reads without fully basecalling them, thereby reducing *both* data movement overheads and overall execution time spent in basecalling and read mapping.

We believe that integrating multiple steps of genome analysis using suitable architectures, such as PIM, can unlock significant opportunities for 1) reducing data movement overheads, 2) eliminating useless basecalling, and 3) avoiding useless data movement and computation in downstream analysis. These approaches have the potential to substantially enhance the performance and energy efficiency of the entire genome analysis pipeline.

3. Accelerating Real-Time Genome Analysis

Real-time genome analysis aims to perform the steps in the genome analysis pipeline (e.g., read mapping) while the raw sequencing data is generated [25, 34]. The main challenges of real-time genome analysis are to 1) match the throughput at which the raw sequencing data is generated, 2) tolerate the noise in the raw sequencing data to provide accurate results, and 3) meet the latency and energy consumption requirements of target applications. Among the HTS technologies, nanopore sequencing is uniquely suited for real-time genome analysis due to its ability to eject reads from nanopores without fully sequencing them, known as *adaptive sampling or Read Until* [25]. This feature can significantly reduce the overall sequencing time and cost and reduce the latency of genome analysis by 1) avoiding full sequencing of reads that will be useless in downstream analysis and 2) overlapping the latency of sequencing with steps in downstream analysis.

To enable real-time genome analysis, several works propose pure algorithmic techniques or algorithm-hardware co-design solutions. First, ReadFish [29], ReadBouncer [134], and RUBRIC [26] use costly basecalling mechanisms for adaptive sampling. These techniques require costly and energy-hungry computational resources. Such a requirement may cause practical challenges in 1) scaling genome analysis to lower energy and cost levels and 2) performing in-the-field sequencing using mobile sequencing devices such as ONT MinION [34]. Second, many works such as UNCALLED [27], Sigmap [28], and RawHash [34] use efficient techniques to utilize adaptive sampling in low-power devices with usually lower accuracy than the basecalling mechanisms. Among these works, RawHash can provide high accuracy for large genomes with an efficient and accurate hash-based similarity identification technique. Third, several algorithm-architecture co-designs use FPGAs [31] or ASICs [121] to provide fast, accurate, and low-power real-time genome analysis. However, these works are applicable only to small genomes, such as viral genomes, as their algorithm designs lack efficient scalability to larger genomes.

We believe that achieving accurate and real-time genome analysis still requires substantial developments in both efficient algorithms and architecture. This can be achieved by 1) designing efficient software that can be used in low-power devices for adaptive sampling and real-time genome analysis, 2) new techniques for genome analysis that do not require translating the raw sequencing data to nucleotide bases, and 3) combining and parallelizing several steps in real-time genome analysis using efficient algorithm-architecture co-designs to minimize the latency (and energy) of time-critical genomics applications.

4. Accelerating Read Mapping

The goal of read mapping is to identify similarities and differences between genomic sequences, such as between a read and a representative sequence of a species, known as a *reference genome*. Due to genomic variants and sequencing errors, differences and similarities between these sequences (i.e., matches, substitutions, insertions, and deletions) are identified using an approximate string matching (ASM) algorithm to generate an *alignment score* that quantifies the degree of similarity between a pair of sequences. This process is known as *sequence alignment*. A pair of sequences is said to be *aligned* when their alignment score shows a sufficiently high degree of similarity. However, ASM algorithms often have quadratic time and space complexity, making them computationally challenging for both long genomic sequences and a large number of sequence pairs. To ease the identification of similarities within vast amounts of sequencing data, read mapping includes multiple steps, such as: 1) sketching [35–40], 2) indexing and seeding [41–45, 47], 3) pre-alignment filtering [46, 48, 49, 76, 90, 135], and 4) sequence alignment (i.e., ASM) [50–55].

Since read mapping is a crucial and computationally expensive step in many genome analysis pipelines, numerous works focus on accelerating it in various ways. First, a significant fraction of sequence pairs do *not* align, which leads to wasted computation and energy during alignment [90]. To avoid this useless computation, several works propose *pre-alignment filtering*, another step in read mapping that can efficiently detect and eliminate highly dissimilar sequence pairs *without* using alignment. Most pre-alignment filtering works [46, 48, 49, 76, 90, 135] provide algorithm-architecture co-design using FPGAs, GPUs, and PIM to substantially accelerate the entire read mapping process by exploiting massive parallelism, efficient bitwise operations, and specialized hardware logic for detecting similarities among a large number of sequences.

Second, GenStore [87] observes that a large amount of sequencing data unnecessarily moves from the solid-state drive (SSD) to memory during read mapping, significantly increasing latency and energy consumption. To eliminate this wasteful data movement, GenStore uses specialized logic *within* the SSD to identify two sets of reads: 1) reads that do not align due to high dissimilarity with the reference genome, and 2) reads that align by exactly

matching the reference genome. Such reads are processed in the storage system and not moved to main memory or the CPU, thereby eliminating unnecessary data movement in the system.

Third, numerous studies, including GenASM [54] and Darwin [117], focus on accelerating the underlying ASM algorithm employed in sequence alignment through efficient algorithm-architecture co-design. They do so by exploiting systolic arrays [115], GPUs [86], FPGAs [115,118,120], ASICs [116], high-bandwidth memory (HBM) [123], and PIM [89,97,105,106]. These works provide substantial speedups of up to several orders of magnitude compared to software baselines. Among these works, SeGraM [123] is the *first* to accelerate aligning sequences to graphs that are used to reduce population bias and improve genome analysis accuracy by representing a large population (instead of a few individuals) within a single reference genome.

Despite recent advancements, read mapping remains a computational bottleneck in genome analysis [18,19]. This is primarily due to the vast amount of sequencing data generated at an ever-increasing rate by sequencing machines, which puts significant pressure on the mapping step due to numerous unnecessary calculations between dissimilar pairs of sequences. Avoiding wasteful 1) data movement, 2) computation, and 3) memory space usage using efficient algorithm-architecture co-design is critical for minimizing the high energy, time, and storage costs associated with read mapping and the entire genome analysis pipeline.

5. Accelerating Variant Calling

The objective of variant calling is to identify genomic variants between an individual's genome and a reference genome [58–64]. These variants are mainly categorized as single-nucleotide polymorphisms (SNPs), insertions, deletions, and larger structural variations (SVs). Accurate and efficient detection of these variants is vital for understanding of the genetic basis of diseases [7], population genetics [63], evolutionary studies [3], personalized medicine [136] and pharmacogenomics [137].

Variant calling involves processing the read mapping output and detecting variants. First, read mapping output is processed by sorting and optionally identifying duplicate information to minimize bias introduced during the *polymerase chain reaction* (PCR) step of sample preparation [138]. Second, mapped reads are analyzed to distinguish genuine variants from sequencing errors or misalignments using resource-intensive statistical techniques [59,61,63] or machine learning techniques [64].

Variant callers like GATK HaplotypeCaller [63] use costly probabilistic calculations to analyze the likelihood of specific variants in large sequencing datasets. DeepVariant [64], a DNN-based variant caller, processes read alignment information as images, demanding substantial GPU resources and memory. Reducing computational requirements through algorithmic optimizations, parallelization, and efficient data representation is crucial for faster, more accurate genetic variant analyses.

To accelerate variant calling, several works propose algorithm-architecture co-designs. These include fast execution of Pair Hidden Markov Models (Pair HMMs) in FPGAs or ASICs [139,140], reducing data movement overheads in GPUs [141], and pipelining processing steps with tools like elPrep [142] and system-on-chip designs [143].

Although several works focus on accelerating variant calling, there is an urgent need for further acceleration, e.g., for DNN-based variant callers that can provide highly accurate results while bypassing certain processing steps, potentially accelerating the entire genome analysis pipeline.

5.1. Analysis of Variants

Following variant calling, it is critical to analyze the identified variants to understand their functional impact on the organism and their role in diseases, population genetics, or evolution. This analysis involves gene annotation [65–69] and enrichment analysis [70–73]. Gene annotation provides relevant information about variants, while enrichment analysis tools identify associations with biological processes, molecular functions, or cellular components. Although these tools need to handle large volumes of data, there is, to our knowledge, little work on accelerating these steps in the genome analysis pipeline. We believe these steps are critical for acceleration using hardware-software co-design.

6. Conclusion and Future Outlook

Rapid advancements in genomic sequencing technologies have led to an exponential increase in generated genomic data. As data generation continues to grow, data movement bottlenecks will increasingly impact performance and waste energy [144,145]. Future research in genome analysis acceleration should focus on at least three main directions. First, addressing data movement and storage challenges is crucial for reducing energy consumption and improving performance. Second, integrating and pipelining multiple genome analysis steps using hardware-software co-design can enhance efficiency by reducing both useless computation and data movement. Third, significant potential exists in enabling accurate and fast real-time genome analysis by co-developing efficient algorithms together with specialized hardware, resulting in low-power, high-performance and cost-effective (portable) sequencing with low latency.

Acknowledgment

We thank all members of the SAFARI Research Group for the stimulating and scholarly intellectual environment they provide. We acknowledge the generous gift funding provided by our industrial partners (especially by Google, Huawei, Intel, Microsoft, VMware). This work was in part supported by the Semiconductor Research Corporation (SRC) and BioPIM.

References

[1] A. Pickar-Oliver and C. A. Gersbach, "The next generation of CRISPR–Cas technologies and applications," *Nat. Rev. Mol. Cell Biol.*, 2019.
[2] T. Shahroodi *et al.*, "Demeter: A Fast and Energy-Efficient Food Profiler Using Hyperdimensional Computing in Memory," *IEEE Access*, 2022.
[3] M. Kanehisa, "Toward understanding the origin and evolution of cellular organisms," *Protein Science*, 2019.
[4] S. Morganti *et al.*, "Next Generation Sequencing (NGS): A Revolutionary Technology in Pharmacogenomics and Personalized Medicine in Cancer," in *Adv. Exp. Med. Biol.*, Cham, 2019.
[5] T. Dunn *et al.*, "SquiggleFilter: An Accelerator for Portable Virus Detection," in *MICRO*, 2021.
[6] M. Alser *et al.*, "COVIDHunter: COVID-19 Pandemic Wave Prediction and Mitigation via Seasonality Aware Modeling," *Front. Public Health*, 2022.
[7] M. S. Lawrence *et al.*, "Mutational heterogeneity in cancer and the search for new cancer-associated genes," *Nature*, 2013.
[8] J. S. Johnson *et al.*, "Evaluation of 16S rRNA gene sequencing for species and strain-level microbiome analysis," *Nat. Comm.*, 2019.
[9] D. R. Bentley *et al.*, "Accurate whole human genome sequencing using reversible terminator chemistry," *Nature*, 2008.
[10] J. Eid *et al.*, "Real-Time DNA Sequencing from Single Polymerase Molecules," *Science*, 2009.
[11] D. Branton *et al.*, "The potential and challenges of nanopore sequencing," *Nat. Biotech.*, 2008.
[12] D. Deamer *et al.*, "Three decades of nanopore sequencing," *Nat. Biotech.*, 2016.
[13] D. Senol *et al.*, "Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions," in *BIB*, 2018.
[14] J. Shendure *et al.*, "DNA sequencing at 40: past, present and future," *Nature*, 2017.
[15] C. Firtina *et al.*, "Apollo: a sequencing-technology-independent, scalable and accurate assembly polishing algorithm," *Bioinform.*, 2020.
[16] M. Alser *et al.*, "Technology dictates algorithms: recent developments in read alignment," *Genome Biol.*, 2021.
[17] Z. D. Stephens *et al.*, "Big Data: Astronomical or Genomical?" *PLOS Biology*, 2015.
[18] M. Alser *et al.*, "Accelerating Genome Analysis: A Primer on an Ongoing Journey," *IEEE Micro*, 2020.
[19] M. Alser *et al.*, "From Molecules to Genomic Variations: Accelerating Genome Analysis via Intelligent Algorithms and Architectures," *CSBJ*, 2022.
[20] R. F. Purnell *et al.*, "Nucleotide Identification and Orientation Discrimination of DNA Homopolymers Immobilized in a Protein Nanopore," *Nano Letters*, 2008.
[21] W. Timp *et al.*, "DNA Base-Calling from a Nanopore Using a Viterbi Algorithm," *Biophysical Journal*, 2012.
[22] V. Boža *et al.*, "DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads," *PLOS One*, 2017.
[23] F. J. Rang *et al.*, "From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy," *Genome Biol.*, 2018.
[24] Z. Xu *et al.*, "Fast-bonito: A faster deep learning based basecaller for nanopore sequencing," *Artificial Intelligence in the Life Sciences*, 2021.
[25] M. Loose *et al.*, "Real-time selective sequencing using nanopore technology," *Nat. Methods*, 2016.
[26] H. S. Edwards *et al.*, "Real-Time Selective Sequencing with RUBRIC: Read Until with Basecall and Reference-Informed Criteria," *Scientific Reports*, 2019.
[27] S. Kovaka *et al.*, "Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED," *Nat. Biotech.*, 2021.
[28] H. Zhang *et al.*, "Real-time mapping of nanopore raw signals," *Bioinform.*, 2021.
[29] A. Payne *et al.*, "Readfish enables targeted nanopore sequencing of gigabase-sized genomes," *Nat. Biotech.*, 2021.
[30] Y. Bao *et al.*, "SquiggleNet: real-time, direct classification of nanopore signals," *Genome Biol.*, 2021.
[31] P. J. Shih *et al.*, "Efficient real-time selective genome sequencing on resource-constrained devices," *arXiv*, 2022.
[32] H. Sadasivan *et al.*, "Rapid Real-time Squiggle Classification for Read Until Using RawMap," *Arch. Clin. Biomed. Res.*, 2023.
[33] A. Senanayake *et al.*, "DeepSelectNet: deep neural network based selective sequencing for oxford nanopore sequencing," *BMC Bioinform.*, 2023.

[34] C. Firtina *et al.*, "RawHash: Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes," in *ISMB*, 2023.
[35] S. Schleimer *et al.*, "Winnowing: Local Algorithms for Document Fingerprinting," in *SIGMOD*, New York, NY, USA, 2003.
[36] M. Roberts *et al.*, "Reducing storage requirements for biological sequence comparison," *Bioinform.*, 2004.
[37] H. Li, "Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences," *Bioinform.*, 2016.
[38] C. Firtina *et al.*, "BLEND: a fast, memory-efficient and accurate mechanism to find fuzzy seed matches in genome analysis," *NARGAB*, 2023.
[39] D. N. Baker and B. Langmead, "Dashing 2: genomic sketching with multiplicities and locality-sensitive hashing," *bioRxiv*, 2023.
[40] A. Joudaki *et al.*, "Aligning distant sequences to graphs using long seed sketches," *Genome Res.*, 2023.
[41] S. F. Altschul *et al.*, "Basic local alignment search tool," *JMB*, 1990.
[42] B. Langmead *et al.*, "Ultrafast and memory-efficient alignment of short DNA sequences to the human genome," *Genome Biol.*, 2009.
[43] H. Xin *et al.*, "Accelerating read mapping with FastHASH," *BMC Genom.*, 2013.
[44] H. Xin *et al.*, "Shifted Hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping," *Bioinform.*, 2015.
[45] H. Xin *et al.*, "Optimal seed solver: optimizing seed selection in read mapping," *Bioinform.*, 2016.
[46] M. Alser *et al.*, "GateKeeper: a new hardware architecture for accelerating pre-alignment in DNA short read mapping," *Bioinform.*, 2017.
[47] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinform.*, 2018.
[48] M. Alser *et al.*, "Shouji: a fast and efficient pre-alignment filter for sequence alignment," *Bioinform.*, 2019.
[49] M. Alser *et al.*, "SneakySnake: a fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs," *Bioinform.*, 2020.
[50] S. B. Needleman and C. D. Wunsch, "A general method applicable to the search for similarities in the amino acid sequence of two proteins," *JMB*, 1970.
[51] T. Smith and M. Waterman, "Identification of common molecular subsequences," *JMB*, 1981.
[52] R. Baeza-Yates and G. H. Gonnet, "A New Approach to Text Searching," *Commun. ACM*, 1992.
[53] G. Myers, "A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming," *J. ACM*, 1999.
[54] D. Senol Cali *et al.*, "GenASM: A High-Performance, Low-Power Approximate String Matching Acceleration Framework for Genome Sequence Analysis," in *MICRO*, 2020.
[55] S. Marco-Sola *et al.*, "Fast gap-affine pairwise alignment using the wavefront algorithm," *Bioinform.*, 2021.
[56] J. S. Kim *et al.*, "AirLift: A Fast and Comprehensive Technique for Remapping Alignments between Reference Genomes," in *APBC*, 2023.
[57] J. S. Kim *et al.*, "FastRemap: A Tool for Quickly Remapping Reads between Genome Assemblies," *Bioinform.*, 2022.
[58] P.-Y. Kwok *et al.*, "Comparative Analysis of Human DNA Variations by Fluorescence-Based Sequencing of PCR Products," *Genomics*, 1994.
[59] D. A. Nickerson *et al.*, "PolyPhred: automating the detection and genotyping of single nucleotide substitutions using fluorescence-based resequencing," *NAR*, 1997.
[60] G. T. Marth *et al.*, "A general approach to single-nucleotide polymorphism discovery," *Nat. Genetics*, 1999.
[61] S. Weckx *et al.*, "novoSNP, a novel computational tool for sequence variation discovery," *Genome Res.*, 2005.
[62] H. Li *et al.*, "Mapping short DNA sequencing reads and calling variants using mapping quality scores," *Genome Res.*, 2008.
[63] R. Poplin *et al.*, "Scaling accurate genetic variant discovery to tens of thousands of samples," *bioRxiv*, 2018.
[64] R. Poplin *et al.*, "A universal SNP and small-indel variant caller using deep neural networks," *Nat. Biotech.*, 2018.
[65] R. Guigó *et al.*, "Prediction of gene structure," *JMB*, 1992.
[66] M. Borodovsky *et al.*, "Detection of new genes in a bacterial genome using Markov models for three gene classes," *NAR*, 1995.
[67] C. Burge and S. Karlin, "Prediction of complete gene structures in human genomic DNA11Edited by F. E. Cohen," *JMB*, 1997.
[68] S. Li *et al.*, "Snap: an integrated SNP annotation platform," *NAR*, 2007.
[69] K. Wang *et al.*, "ANNOVAR: functional annotation of genetic variants from high-throughput sequencing data," *NAR*, 2010.
[70] M. Ashburner *et al.*, "Gene Ontology: tool for the unification of biology," *Nat. Genetics*, 2000.
[71] S. W. Doniger *et al.*, "MAPPFinder: using Gene Ontology and GenMAPP to create a global gene-expression profile from microarray data," *Genome Biol.*, 2003.
[72] A. Subramanian *et al.*, "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles," *PNAS*, 2005.
[73] B. Otlu *et al.*, "GLANET: genomic loci annotation and enrichment tool," *Bioinform.*, 2017.
[74] L. G. Biesecker and R. C. Green, "Diagnostic Clinical Genome and Exome Sequencing," *NEJM*, 2014.
[75] S. Marco-Sola *et al.*, "Optimal gap-affine alignment in O(s) space," *Bioinform.*, 2023.
[76] Z. Bingöl *et al.*, "GateKeeper-GPU: Fast and Accurate Pre-Alignment Filtering in Short Read Mapping," in *IPDPSW*, 2021.
[77] M. B. Cavlak *et al.*, "TargetCall: Eliminating the Wasted Computation in Basecalling via Pre-Basecalling Filtering," in *APBC*, 2023.
[78] G. Singh *et al.*, "A Framework for Designing Efficient Deep Learning-Based Genomic Basecallers," *bioRxiv*, 2022.
[79] F. Hach *et al.*, "mrsFAST: a cache-oblivious algorithm for short-read mapping," *Nat. Methods*, 2010.
[80] T. Pan *et al.*, "Kmerind: A Flexible Parallel Library for K-Mer Indexing of Biological Sequences on Distributed Memory Systems," in *BCB*, 2016.
[81] M. Ellis *et al.*, "DiBELLA: Distributed Long Read to Long Read Alignment," in *ICPP*, 2019.
[82] J. Daily, "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments," *BMC Bioinform.*, 2016.
[83] M. Vasimuddin *et al.*, "Efficient architecture-aware acceleration of BWA-MEM for multicore systems," in *IPDPS*, 2019.
[84] G. Singh *et al.*, "FPGA-based Near-Memory Acceleration of Modern Data-Intensive Applications," *IEEE Micro*, 2021.
[85] S. Kalikar *et al.*, "Accelerating minimap2 for long-read sequencing applications on modern CPUs," *Nat. Comput. Sci.*, 2022.
[86] J. Lindegger *et al.*, "Scrooge: A Fast and Memory-Frugal Genomic Sequence Aligner for CPUs, GPUs, and ASICs," *Bioinform.*, 2023.
[87] N. Mansouri Ghiasi *et al.*, "GenStore: A High-Performance in-Storage Processing System for Genome Sequence Analysis," in *ASPLOS*, 2022.
[88] H. Mao *et al.*, "GenPIP: In-Memory Acceleration of Genome Analysis via Tight Integration of Basecalling and Read Mapping," in *MICRO*, 2022.
[89] R. Kaplan *et al.*, "A resistive CAM processing-in-storage architecture for DNA sequence alignment," *IEEE Micro*, 2017.
[90] J. S. Kim *et al.*, "GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies," *BMC Genom.*, 2018.
[91] R. Kaplan *et al.*, "RASSA: Resistive pre-alignment accelerator for approximate DNA long read mapping," *IEEE Micro*, 2018.
[92] S. Gupta *et al.*, "RAPID: A reRAM processing in-memory architecture for DNA sequence alignment," in *ISLPED*, 2019.
[93] S. Angizi *et al.*, "AlignS: A processing-in-memory accelerator for DNA short read alignment leveraging SOT-MRAM," in *DAC*, 2019.
[94] S. Ghose *et al.*, "The Processing-in-Memory Paradigm: Mechanisms to Enable Adoption," in *Beyond-CMOS Technologies for Next Generation Computer Design*, 2019.
[95] S. Angizi *et al.*, "Exploring DNA alignment-in-memory leveraging emerging SOT-MRAM," in *GLSVLSI*, 2020.
[96] Q. Lou *et al.*, "Helix: Algorithm/architecture co-design for accelerating nanopore genome base-calling," in *PACT*, 2020.
[97] F. Chen *et al.*, "PARC: A processing-in-CAM architecture for genomic long read pairwise alignment using ReRAM," in *ASP-DAC*, 2020.
[98] Z. I. Chowdhury *et al.*, "A DNA read alignment accelerator based on computational RAM," *JXCDC*, 2020.
[99] S. Angizi *et al.*, "PIM-Aligner: A processing-in-MRAM platform for biological sequence alignment," in *DATE*, 2020.
[100] R. Kaplan *et al.*, "BioSEAL: In-memory biological sequence alignment accelerator for large-scale genomic data," in *SYSTOR*, 2020.
[101] A. F. Laguna *et al.*, "Seed-and-Vote based in-memory accelerator for DNA read mapping," in *ICCAD*, 2020.
[102] S. K. Khatamifard *et al.*, "GeNVoM: Read mapping near non-volatile memory," *TCBB*, 2021.
[103] M. Khalifa *et al.*, "FiltPIM: In-memory filter for DNA sequencing," in *ICECS*, 2021.
[104] X.-Q. Li *et al.*, "PIM-Align: A processing-in-memory architecture for FM-Index search algorithm," *JCST*, 2021.
[105] S. Diab *et al.*, "High-throughput Pairwise Alignment with the Wavefront Algorithm using Processing-in-Memory," in *IPDPSW*, 2022.
[106] S. Diab *et al.*, "A Framework for High-throughput Sequence Alignment using Real Processing-in-Memory Systems," *arXiv*, 2022.
[107] A. Madhavan *et al.*, "Race Logic: A Hardware Acceleration for Dynamic Programming Algorithms," in *ISCA*, 2014.
[108] P. Chen *et al.*, "Accelerating the next generation long read mapping with the FPGA-based system," *TCBB*, 2014.
[109] H. M. Waidyasooriya and M. Hariyama, "Hardware-acceleration of short-read alignment based on the Burrows-wheeler transform," *TPDS*, 2015.
[110] Y.-T. Chen *et al.*, "A novel high-throughput acceleration engine for read alignment," in *FCCM*, 2015.
[111] Y.-T. Chen *et al.*, "When Spark Meets FPGAs: A case study for next-generation DNA sequencing acceleration," in *HotCloud*, 2016.
[112] A. Goyal *et al.*, "Ultra-fast Next Generation Human Genome Sequencing Data Processing Using DRAGENTM Bio-IT Processor for Precision Medicine," *Open J. Genet.*, 2017.
[113] S. S. Banerjee *et al.*, "ASAP: Accelerated short-read alignment on programmable hardware," *TC*, 2019.
[114] E. Rucci *et al.*, "SWIFOLD: Smith-Waterman implementation on FPGA with OpenCL for long DNA sequences," *BMC Syst. Biol.*, 2018.
[115] X. Fei *et al.*, "FPGASW: Accelerating large-scale Smith–Waterman sequence alignment application with backtracking on FPGA linear systolic array," *Interdiscip. Sci.*, 2018.
[116] D. Fujiki *et al.*, "Genax: A Genome Sequencing Accelerator," in *ISCA*, 2018.
[117] Y. Turakhia *et al.*, "Darwin: A Genomics Co-processor Provides up to 15,000 x Acceleration on Long Read Assembly," in *ASPLOS*, 2018.
[118] D. Fujiki *et al.*, "SeedEx: A genome sequencing accelerator for optimal alignments in subminimal space," in *MICRO*, 2020.
[119] Z. Wu *et al.*, "FPGA-accelerated 3rd generation DNA sequencing," *TBCS*, 2020.
[120] A. Haghi *et al.*, "An FPGA accelerator of the wavefront algorithm for genomics pairwise alignment," in *FPL*, 2021.
[121] T. Dunn *et al.*, "SquiggleFilter: An accelerator for portable virus detection," in *MICRO*, 2021.
[122] C. Firtina *et al.*, "ApHMM: A Profile Hidden Markov Model Acceleration Framework for Genome Analysis," *arXiv*, 2022.
[123] D. Senol Cali *et al.*, "SeGraM: A Universal Hardware Accelerator for Genomic Sequence-to-Graph and Sequence-to-Sequence Mapping," in *ISCA*, 2022.
[124] A. Cacho *et al.*, "A Comparison of Base-calling Algorithms for Illumina Sequencing Technology," *Briefings Bioinform.*, 2016.
[125] A. M. Wenger *et al.*, "Accurate circular consensus long-read sequencing improves variant detection and assembly of a human genome," *Nat. Biotechnol.*, 2019.
[126] C.-S. Chin *et al.*, "Nonhybrid, finished microbial genome assemblies from long-read SMRT sequencing data," *Nat. Methods*, 2013.
[127] X. Lv *et al.*, "An end-to-end Oxford nanopore basecaller using convolution-augmented transformer," in *BIBM*, 2020.
[128] J. Zeng *et al.*, "Causalcall: Nanopore basecalling using a temporal convolutional network," *Frontiers in Genetics*, 2020.
[129] H. Konishi *et al.*, "Halcyon: an accurate basecaller exploiting an encoder-decoder model with monotonic attention," *Bioinform.*, 2021.
[130] P. Perešíni *et al.*, "Nanopore base calling on the edge," *Bioinform.*, 2021.
[131] Y.-M. Yeh and Y.-C. Lu, "MSRCall: A multi-scale deep neural network to basecall Oxford nanopore sequences," *Bioinform.*, 2022.
[132] N. Huang *et al.*, "SACall: A neural network basecaller for Oxford nanopore sequencing data based on self-attention mechanism," *TCBB*, 2022.
[133] Q. Lou and L. Jiang, "Brawl: A spintronics-based portable basecalling-in-memory architecture for nanopore genome sequencing," *CAL*, 2018.
[134] J.-U. Ulrich *et al.*, "ReadBouncer: precise and scalable adaptive sampling for nanopore sequencing," *Bioinform.*, 2022.
[135] M. Alser *et al.*, "MAGNET: Understanding and Improving the Accuracy of Genome Pre-Alignment Filtering," *arXiv*, 2017.
[136] Y. Dong *et al.*, "Genome-Wide Off-Target Analysis in CRISPR-Cas9 Modified Mice and Their Offspring," *G3*, 2019.
[137] K. Sangkuhl *et al.*, "Pharmacogenomics Clinical Annotation Tool (PharmCAT)," *Clin. Pharm. Therap.*, 2020.
[138] S. Zverinova and V. Guryev, "Variant calling: Considerations, practices, and developments," *Human Mutation*, 2022.
[139] S. Ren *et al.*, "FPGA acceleration of the pair-HMMs forward algorithm for DNA sequence analysis," in *BIBM*, 2015.
[140] X. Wu *et al.*, "A High-Throughput Pruning-Based Pair-Hidden-Markov-Model Hardware Accelerator for Next-Generation DNA Sequencing," *IEEE Solid-State Circuits Letters*, 2021.
[141] E. Li *et al.*, "Improved GPU Implementations of the Pair-HMM Forward Algorithm for DNA Sequence Alignment," in *ICCD*, 2021.
[142] C. Herzeel *et al.*, "Multithreaded variant calling in elPrep 5," *PLOS ONE*, 2021.
[143] Y.-C. Wu *et al.*, "A 975-mW Fully Integrated Genetic Variant Discovery System-on-Chip in 28 nm for Next-Generation Sequencing," *IEEE J. Solid-State Circuits*, 2021.
[144] G. F. Oliveira *et al.*, "DAMOV: A New Methodology and Benchmark Suite for Evaluating Data Movement Bottlenecks," *IEEE Access*, 2021.
[145] O. Mutlu *et al.*, "A Modern Primer on Processing in Memory," in *Emerging Computing: From Devices to Systems: Looking Beyond Moore and Von Neumann*, 2023.