The Genomic Computing Revolution: Defining the Next Decades of Accelerating Genomics

Harisankar Sadasivan
AI Group
Advanced Micro Devices Inc.
Bellevue, WA, USA
hsadasiv@amd.com

Artur Klauser
Seattle, WA, USA
Artur.Klauser@computer.org

Juergen Hench
Institute for Pathology and Medical Genetics
University Hospital Basel
Basel, Switzerland
Juergen.Hench@usb.ch

Yatish Turakhia
Electrical and Computer Engineering
University of California
San Diego, USA
yturakhia@ucsd.edu

Gagandeep Singh
Research and Development
Advanced Micro Devices Inc.
Zug, Switzerland
Gagandeep.Singh@amd.com

Alberto Zeni
AI Group- Silicon
Advanced Micro Devices Inc.
Dublin, Ireland
Alberto.Zeni@amd.com

Sarah Beecroft
Pawsey Supercomputing Research Centre
Perth, WA, Australia
Sarah.Beecroft@pawsey.com.au

Satish Narayanasamy
Computer Science and Engineering
University of Michigan Ann Arbor
MI, USA
nsatish@umich.edu

Jeff Nivala
Computer Science and Engineering
University of Washington
Seattle, WA, USA
jmdn@cs.washington.edu

Bob Robey
HPC Solutions, DCGPU
Advanced Micro Devices Inc.
Los Alamos, NM, USA
Bob.Robey@amd.com

Onur Mutlu
Information Technology and Electrical Engineering
ETH Zürich
Zürich, Switzerland
onur.mutlu@inf.ethz.ch

Kristof Denolf
Research and Development
Advanced Micro Devices Inc.
Longmont, CO, USA
kristof.denolf@amd.com

Sriranjani Sitaraman
HPC Solutions, DCGPU
Advanced Micro Devices Inc.
Austin, TX, USA
gina.sitaraman@amd.com

Abstract*—The rapid growth of genomic data has positioned genomics to become one of the world's most storage-intensive and computationally demanding fields. This review paper provides a comprehensive analysis of emerging trends in software and hardware for genomics, addressing a critical gap in the current literature. We categorize and examine key algorithms in genomic sequencing workflows, including sequence data indexing, retrieval and pile-up, dynamic programming, and machine learning. We analyze how computational and memory requirements scale with input sizes for each category, offering insights into potential bottlenecks and optimization opportunities. Our review synthesizes recent advancements in hardware-software co-design, compression schemes, and acceleration techniques for genomic data processing. We propose graphics processing units (GPUs) as a promising first step for deploying genomics workflows due to their high throughput, memory bandwidth, and programmability. By providing a holistic perspective on the computational challenges in genomics, this paper attempts to set a promising direction for future researchers to work collectively to optimize all the components of genomic data processing workflows. Our work aims to foster more targeted and effective advancements in the field, potentially leading to significant improvements in the efficiency and scalability of genomic analyses. We also discuss performance portability as a second step to meet the diverse requirements in point-of-care workflows.

Keywords—Dynamic Time Warping, Dynamic Programming, Genomics, GPU, Illumina, MinION, Machine Learning, Nanopore, PacBio, Illumina

I. INTRODUCTION

As we navigate through the 21st century, genomics is undergoing a significant transformation. Once primarily a research domain, genomics is now emerging as a critical component of healthcare [1]-[3], [80], thanks to the decreasing costs of sequencing [75] and its increasing relevance in disease diagnosis, treatment, and prevention. Sequencing data is now used in various applications, including cancer, rare genetic diseases, microbiome, metagenomics, epidemiology, species conservation, evolutionary biology, crop genomics, and population genomics [87]. This evolution, accompanied by new funding avenues [109] and larger participant cohorts, is leading to a rapid surge in genomic data that demands substantial storage capacity [4], [6], [48], [103]. Further, with the break-out of SARS-CoV-2, we saw an explosion in the amount of sequencing data generated [74] that overwhelmed existing bioinformatics infrastructures [89]. Realizing the importance of genomics, the US Government has categorized biotechnologies (of which genomics is a part) and advanced computing as two of the few critical and emerging technologies (CET) significant to national security [5].

The Global Alliance for Genomics and Health (GA4GH) [6], [48] anticipates that by 2025, approximately 60 million human genomes will have been sequenced globally, with healthcare accounting for the majority. An experiment to sequence a whole human genome generates approximately 30 times more data than the human diploid (double-stranded) DNA, depending on the sequencing depth (coverage) required to account for sequencing errors. While theoretically, a single diploid genome of 6 billion bases could be stored using 1.5 GB `requirements are significantly higher. When combined with

©2024 IEEE

various types of metadata, such as sequence names and quality scores, the annual data storage requirement could range between 2 to 40 exabytes [6].

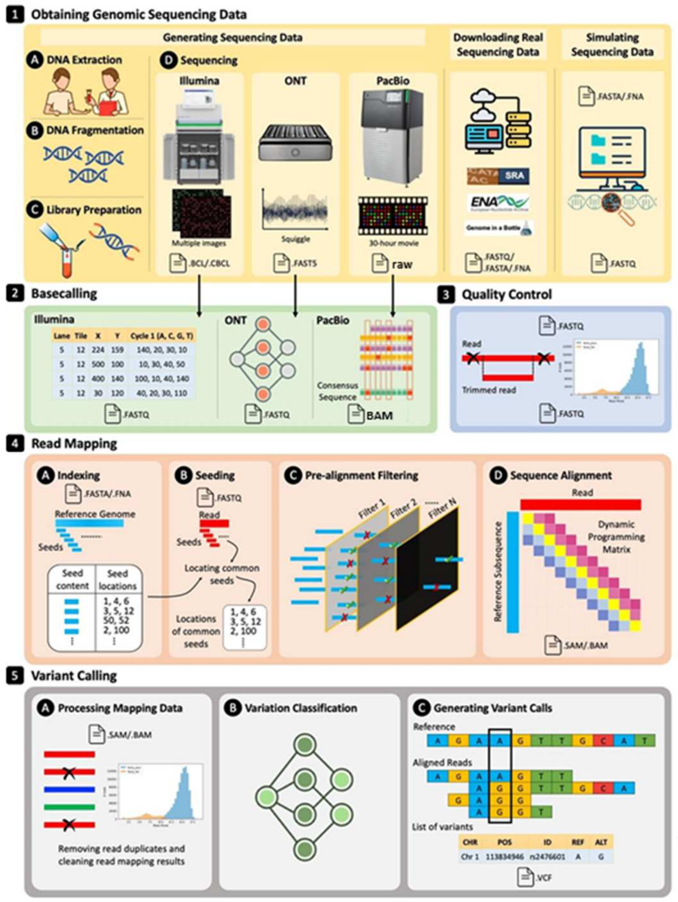
When compared to other data-intensive fields — astronomy generating 1 exabyte/year, particle physics producing over 75 petabytes/year, and YouTube creating 1-2 exabytes/year — genomics is expected to surpass these historical data generators [4], [6]. Consequently, genomics is on the verge of becoming one of the world's most storage-intensive workloads, posing a significant compute, storage, and input/output (I/O) challenge. This impending data avalanche underscores the pressing need for innovative solutions to efficiently manage, store, and analyze genomic data.

Addressing the storage demands of the vast volume of genomic data necessitates innovative solutions. One such approach involves the application of genomic data compression schemes that leverage the similarity that many genomic sequences share due to evolutionary relationships [7]-[9], [90], [91], [101] and are specifically tailored for high-performance computing (HPC) environments. These schemes offer a potential solution to decrease the storage requirements.

Additionally, denser data storage mediums, such as DNA-based storage, could be explored. This approach, however, will necessitate further research to enhance the computational efficiency (reliability and latency) of molecular read/write operations [25], [26], [77], [102]. An intriguing consideration is the point at which it may be more practical to discard the digital genomic sequencing data and preserve the original DNA of the genome [102]. Given advancements in DNA storage technology, retaining the physical DNA could serve as a more efficient method of storing genomic information, allowing for future sequencing and analysis as needed while significantly reducing digital storage demands.

Another strategy is limiting the generation of redundant data. This could be achieved by implementing filtering mechanisms or adopting selective sequencing techniques [14], [22], [27]-[30]. However, this approach presents its own challenges, particularly given the high throughput of real-time sequencers like Oxford Nanopore Technologies (ONT), which require substantial computational power to make real-time decisions while sequencing. We estimate that the largest sequencers from PacBio and ONT would generate raw data on the order of several Giga-bytes per second. It is also worth noting that sometimes for ONT, this huge raw data dump is preserved to retain metadata on methylation, acetylation, and other base modifications depending on the application. This data deluge is reduced by one to three orders of magnitude in size during the basecalling step, depending on the sequencing technology used. Keeping up with this throughput in real-time is challenging for both the software [13], [28], [29] and hardware [13], [104].

Addressing these computational challenges is no easy task. As shown in Fig. 1, genomic data often undergoes a series of algorithmic processes to denoise the signal and reach a consensus on its representation [19]. These processes frequently employ techniques rooted in classical signal processing, index lookups, dynamic programming (DP), statistical analysis, and, more recently, machine learning (ML), including the use of transformer models [18], [19]. Recently, several attempts have been made to understand these algorithms at a higher level by building benchmark suites for different hardware [18], [19]. While there have been numerous attempts [10]-[17], [20]-[24], [27]-[30], [59], [60], [82], [103]-[108] to accelerate these individual components, there is no clear direction set on the hardware of choice as the independent development and constant evolution of these components present a significant challenge. We propose GPUs as an initial platform of choice for deploying genomics workflows because of their high throughput (100s of TFLOPs), memory bandwidth (several TB/s), and, more importantly, programmability. Moreover, many components in genomics processing workflows have been independently ported to GPUs [23], [57], [62], [65], [86], [105].



Although GPUs may currently provide a compelling platform, we extend our discussion to performance portability

[1] Image is adapted from [80] with license: https://creativecommons.org/licenses/by/4.0/. Please note that the output file formats of the PacBio sequencer and basecaller are modified for correctness.

frameworks, acknowledging that point-of-care testing (POCT) genomics workflows present a unique set of challenges [31]-[33], [79] that are distinct from the traditional HPC and, hence, may have a different set of preferences. Data privacy [32] concerns in some government jurisdictions necessitate local processing, while wet lab environments impose strict limitations on size [31], heat generation [33], and noise [79]. These workflows also require high reliability with minimal downtimes. Edge computing architectures address these challenges by bringing computation closer to data sources [88]. Limited storage and bandwidth underscore the need for efficient on-the-fly data compression and decompression, demanding specialized software and hardware support. These requirements are shaping the development of tailored solutions for decentralized genomic data processing, balancing the needs for rapid analysis, data security [34], and operational constraints in clinical settings. In the future, as collaborative genome analysis using privacy-preserving techniques like confidential computing for genomics [76] becomes more prevalent, edge computing on secure platforms [78] may become the standard. This could allow for more widespread and secure genomic data sharing and analysis while maintaining strict privacy protections.

Having established the diverse demands driving the genomics compute and storage tidal wave, we look ahead to guide future genomics workflow development efforts in choosing hardware (e.g., GPUs) that can not only meet the high compute and memory demands but also enable easy programmability and performance portability. We discuss how some of these algorithms may be mapped onto the GPUs.

II. Algorithms In Sequencing

We categorize the most important algorithms that are used in genomics workflows into three categories and explain how the memory and compute requirements scale with input sizes for each of them.

A. Sequence Data Indexing, Retrieval and Pile-up

In genomics, efficient sequence search is a critical operation, and two important techniques for k-mer lookup are hash table lookup [35], [36] and full-text index in minute space (FM) index [37]-[39] lookup. Hash table lookup is commonly used for quick access to fixed-length k-mer occurrences. In this method, a hash function maps each reference k-mer to a unique index in a hash table. The hash table stores the k-mers as keys and their corresponding positions or counts as values. During lookup, the hash function is applied to the query k-mer to determine its hash value and retrieve the associated information from the hash table.

De Bruijn graph traversal [40]-[46], a common operation in genome assembly and variant calling, also frequently employs lookup tables. The bandwidth and compute requirements for the De Bruijn graph traversal scale with the size of the input data and the complexity of the graph. As the number of k-mers increases, both the memory footprint of the graph and the computational cost of traversing it grow. The branching factor of the graph, which is influenced by factors such as sequencing errors and genomic variations, can significantly impact the computational complexity of graph traversal algorithms. Parallel k-mer extraction and counting, compact k-mer representations, sparse matrix representations, and parallel graph traversals are some of the techniques used to accelerate De Bruijn graphs on GPUs.

However, hash tables can suffer from collisions, and irregular access patterns can lead to cache misses, impacting performance [47]. To address these issues, learned indexes like the recursive model index (RMI) have been proposed as a potential solution [47]. RMI employs a two-layer structure: the root node determines which leaf node in the second layer a query maps to, while in the second layer, subsets of k-mers are sorted (an important software preprocessing step) and stored in leaf nodes to optimize lookup times. In case of mispredictions, last-mile lookup is often fast. The bandwidth and compute requirements for RMI scale with the index size. As the number of k-mers increases, both the memory footprint and the computational complexity of traversing the index grow. However, the hierarchical nature of RMI can help mitigate some of this scaling, as it allows for more efficient navigation of large datasets than traditional hash tables. Although RMI for seeding (finding short exact k-mer matches) is currently implemented on CPUs [47], we observe that it has the potential to be effectively parallelized on GPUs using techniques such as privatization and constant memory caching. Additionally, GPUs, with their wider buses and use of high bandwidth memory (HBM), offer higher memory bandwidths [110] that can benefit this memory-bound workload.

FM index lookup is another technique preferred for efficient substring search operations in large genomic sequences which is based on the Burrows-Wheeler transform (BWT). This method involves constructing the FM index from the reference sequence and then searching for a query k-mer within the index. The FM index offers advantages in terms of space efficiency and the ability to perform inexact matching. However, FM index can be more computationally intensive because construction and querying are memory-bound due to pseudo-random accesses. Several works have explored techniques to reduce memory accesses using compact FM indexes, K-step FM index lookups, and thread cooperative approaches [49].

Pile-up is a critical pre-processing operation in variant calling [63], [64], [66], and consensus generation. It involves aligning multiple sequenced reads to a reference genome and then analyzing the aligned bases at each genomic position, generating counts for different bases, insertions, and deletions. Its time complexity is O(LD), where L is the length of the genomic region, and D is the sequencing depth. Memory requirements are substantial, with a space complexity of O(L + LD), often increased by additional optimization structures.

A key challenge in pile-up operations is managing irregular memory access patterns, particularly problematic on GPUs where memory coalescing is crucial for performance. Reads stored in compressed formats spanning different memory pages can lead to cache misses and reduced efficiency.

To address these issues, several optimization strategies have been proposed:

- Binning [66] reads by genomic position to improve memory locality

- Employing compression-aware algorithms [72] to reduce memory bandwidth requirements
- Utilizing pre-computed lookup tables for common operations [111]
- Using custom-designed hardware on FPGAs [113-114] and ASICs [111-112]
- Near-data [117-118] and in-memory processing [115-116] techniques which often require modifications to the DRAM die
- Leveraging parallel processing on GPUs [66]

We note that index table lookup and pileup are typically memory-bound and could benefit from cache-friendly data structures and access patterns, larger caches (shared memory, constant memory), and hardware-supported asynchronous cache prefetching, in addition to all the software optimizations discussed.

B. Dynamic Programming (DP)

DP [92] is a powerful computational technique that can solve complex problems by breaking them down into simpler subproblems. This paradigm is often exploited in bioinformatics for sequence alignment, gene finding [35], variant calling [44]-[46], phylogenetics, and RNA structure prediction [92]. Here, we will discuss the application of 1-D and 2-D DP in various bioinformatics algorithms. In bioinformatics, 1-D DP is often used in the seed chaining [35], [43], [45] process, a common step in genome alignment. Its goal is to find the optimal chain of seeds, which are exact matches between two sequences, which can then be used as a starting point for sequence alignment. This process' compute and memory complexity varies from nlog(n) to n^2, depending on the specific chaining algorithm [47]. However, the workload size is often relatively irregular, making high hardware utilization challenging. Further, algorithmic transformations may be required to reduce divergent memory access and expose more parallelism in the chaining computation [23], [24].

In 2-D DP algorithms, developers often exploit a wavefront parallelism along the diagonal of the matrix used to compute the algorithms. For Smith-Waterman (SW) [51], [52] and Needleman-Wunsch (NW) [50] algorithms, this technique is used for base-level alignment of sequences; this involves filling a matrix of size m x n (where m and n are the lengths of the two aligned sequences). Dynamic time warping (DTW) [14], [22], on the other hand, is used to align raw signals, and pair-hidden Markov models (pair-HMM) [41] are used in probabilistic sequence alignment. For SW, NW, DTW, and pair-HMM, the time complexity is O(mn), and the space complexity is O(mn), where m and n are the lengths of the sequences. Partial Order alignment [46], [53], [54] used for multiple sequence alignment has a time and space complexity of O(mn), where n is the number of sequences and m is the total length of all sequences. The bit-parallel Myers algorithm [55] used for approximate string matching has a time complexity of O(mn/w), where w is the word size of the machine.

Wavefront alignment (WFA) [56] uses 2-D DP with a time complexity of O(ns) using $O(s^2)$ memory, where n is the read length and s is the alignment score. WFA is similar to Myers but only computes a minimum number of DP cells along the diagonal of the matrix using an adaptive band. Its computation achieves the same accuracy as other 2-D alignment algorithms, while its runtime is highly dependent on the sequences' similarity.

Adaptive banded event alignment (ABEA) [57] compares raw signals to a reference genome and employs an adaptive band to capture long gaps in raw signals. ABEA also uses a different recurrence algorithm called Suzuki-Kasahara (SK) [58], which follows the same paradigm as SW and NW but with the added benefit of decreasing register usage.

DP can be made compute-bound on GPUs by employing techniques to fetch fewer data for the same amount of compute. Prior works have explored various software optimizations such as shared memory usage, cache-friendly data structures [60], bit-parallel compute optimizations, using heuristics to reduce search space [57], and various fine-grained (wavefront-level) and coarse-grained (block-level) parallelism strategies [59].

On the hardware front, we note that improved hardware support for wavefront register shuffles (for inter-thread communication), larger and distributed shared memory buffering between blocks, and instructions [83] for reduction operations like min, max, add, etc., could potentially improve the performance of DP-based workloads on GPUs.

C. Machine Learning (AML)

The integration of ML, particularly deep learning techniques, has recently emerged as a powerful tool in genomics workflows. Transformers [61], a type of neural network architecture originally designed for natural language processing, shows remarkable potential in various genomic applications and empowered genomic foundation models [93], [94].

One prominent application is in basecalling, the process of converting raw electrical signals from nanopore sequencing into DNA sequences. Basecalling is the bottleneck step in sequencing workflows [13], and conventional methods often struggle with accuracy, especially in homopolymer regions [67]. ML-based approaches, such as ONT's Dorado [62] basecaller, which uses a modified transformer architecture, have significantly improved accuracy and speed [81]. The compute requirements for these models scale with the input signal length and the model size, typically $O(n^2)$, for self-attention operations where n is the sequence length.

In variant calling, deep learning models [63]-[66] are being employed to improve the accuracy of detecting genomic variations. For instance, DeepVariant [65] uses convolutional neural networks (CNNs) to analyze pileup images of aligned reads. The compute and memory requirements for these models scale with the number of candidate variant sites and the depth of sequencing coverage.

The use of these ML models introduces new computational challenges, especially for the fixed compute resources available onboard a sequencing instrument. They often require significant memory bandwidth for weight retrieval and intermediate activations. The compute requirements are substantial, with matrix multiplications dominating the workload. However, these operations are highly parallelizable, making GPUs well-suited for their acceleration. GPU backend

ML libraries for generalized matrix-matrix multiplication (GEMMs) [68], [69] are optimized for better cache blocking using shared-memory-optimized hierarchical GEMM implementations and stream-k for uniform work partitioning. While the prefill (prompt processing) phase uses compute-bound GEMMs, the autoregressive decode (token generation) phase uses memory-bound generalized matrix-vector multiplication (GEMV). The decode phase may be made compute-bound using several strategies, such as paged-attention (using vLLMs) [70] and grouped query attention (GQA) [71].

As model sizes grow, the memory footprint becomes a critical factor. For example, a typical transformer model for basecalling might require several gigabytes of memory for weights alone. This scales with the square of the model's hidden dimension size. The input sequence length also affects memory usage, as attention mechanisms typically require $O(n^2)$ memory for sequences of length n.

In terms of compute scaling, transformer models generally have a time complexity of $O(n^2d)$, where n is the sequence length and d is the hidden dimension size. This quadratic scaling with sequence length can become problematic for long reads, necessitating techniques like sliding window attention or linear attention variants.

Moreover, current basecallers use high-precision datatypes (e.g., 32 bits) to represent each neural network layer present in a basecaller. This leads to high bandwidth and processing demands [84]. Thus, current basecallers with high arithmetic precision have inefficient hardware implementations.

The integration of these ML models into genomics pipelines presents both opportunities and challenges. While they offer improved accuracy and the potential to uncover complex patterns, they also introduce significant computational demands. Token generation phase and CNNs with small filter masks (as in Deepvariant) can benefit from better hardware support for grouped GEMV multiplication in GPU's tensor/matrix cores and better cache prefetching mechanisms. Hardware and software support for newly emerging low-precision micro-scaling data formats could help processing efficiency as well [85]. Additionally, sparsity support in software and hardware could help improve any data movement bottlenecks.

III. Performance Portability

The diverse requirements at the point-of-care testing (POCT) compared to the traditional HPC for genomics necessitate a move towards performance portability. While software development on GPUs as the first step offers significant performance improvements for many genomics algorithms, the field can benefit from solutions adaptable to various hardware architectures. In this context, it's worth discussing the role of domain-specific languages (DSLs) [73], [99] and software libraries [96]-[98] which have made significant contributions in fields like machine learning. They enable backend compiler optimizations and high-level synthesis (HLS) frameworks [100] for a broad range of hardware accelerators. Similar approaches are conceivable for genomic workloads, particularly for computationally intensive tasks like dynamic programming algorithms [95].

As the second step, to enable edge processing, we propose using a CPU-GPU SoC or using the DSL SYCL [73] to move to FPGAs, as FPGAs meet most of the requirements at the edge. We propose a two-step approach to achieve performance portability in genomics workflows, starting with GPUs.

The first step involves optimizing genomics workflows for GPUs. GPUs offer several advantages. GPUs provide massive parallelism and high memory bandwidth, offering significant speedups for all the various components in genomics workflows. GPUs are easily programmable, and many individual components in the genomics workflow are already ported to GPUs [23], [57], [62], [65], [86]. Key optimization strategies for GPUs include efficient data streaming techniques to handle large genomic datasets, coalesced memory access and shared memory utilization, dynamic work distribution, adaptive load balancing, and redesigning algorithms to expose more parallelism. Further, adapting algorithms for GPU execution often reveals opportunities for parallelization and optimization that might not be apparent in CPU-only implementations.

Genomics-specific operations in point-of-care testing (POCT) have strict requirements for low energy consumption, heat emission, and lab space utilization. As a solution to this, running the GPU workflows on edge computing solutions consisting of CPU-GPU SoCs (with unified memory on board) is an easy option. Alternatively, once GPU optimizations are in place, developers could leverage DSLs like SYCL [73] to extend their implementations to FPGAs. SYCL is an open standard that enables single-source heterogeneous programming for various accelerators, including FPGAs. It allows developers to write code once and target multiple platforms. Using SYCL to run on FPGAs offers several benefits. Much of the GPU-optimized code can be reused, significantly reducing development time. SYCL provides high-level abstractions that can be efficiently mapped to FPGA resources. Additionally, FPGAs excel at custom datapath creation for genomics-specific operations in point-of-care testing (POCT).

However, this approach also presents challenges. While SYCL provides portability, achieving optimal performance on FPGAs typically requires architecture-specific optimizations. FPGA implementations need careful consideration of resource usage (look-up tables, digital signal processing blocks, memory blocks), which differs from GPU optimization. Moreover, FPGA synthesis and place-and-route processes can be time-consuming compared to GPU compilation, which may impact development cycles.

By following this two-step approach, genomics researchers and developers can create high-performance, portable implementations of their workflows. This strategy allows them to leverage the immediate benefits of GPU acceleration while paving the way for future adaptability to FPGAs and other emerging architectures. As the field of genomics continues to evolve, such performance-portable solutions will be crucial in ensuring that computational methods can keep pace with the growing demands of genomic data analysis.

IV. Prior Work

Extensive research has been conducted on various aspects of genomic data processing and storage. Significant advancements have been made in utilizing DNA as a storage medium [25], [26], [77], developing selective sequencing technologies [14], [22], [27]-[30], and creating efficient compression schemes [7]-[9] for genomic data storage. Concurrently, numerous studies have focused on hardware-software co-design [10]-[17], [20], [21], [82], [104], [106], in-memory or near-data processing [115-118] and software-only [22], [24], [27]-[30], [59], [60], [107], [108] optimizations to accelerate individual components within genome processing workflows.

Recent literature has also seen the emergence of genomics benchmarks [18], [19] and their evaluation on diverse hardware platforms. However, a critical gap remains in the existing body of work. There is a notable absence of comprehensive studies that synthesize the compute and memory requirements across the spectrum of genomic workloads. This lack of a holistic perspective hinders the ability to identify overarching trends and establish a unified direction for optimizing various components of genomic processing pipelines.

The integration of ML, particularly deep learning techniques and transformers, into genomics workflows introduces new computational challenges that have not been fully addressed in the context of end-to-end genomic analysis pipelines. Our work addresses this gap by providing a systematic analysis of the computational and memory demands of these workloads. We propose using GPUs as the primary platform for genomics workflows, followed by a transition to performance-portable solutions, representing a novel strategy that aims to leverage immediate benefits while preparing for the evolving landscape of computing hardware in genomics.

V. Discussion

The exponential growth of genomic data necessitates a shift towards more powerful and flexible computing solutions. Our review emphasizes the importance of starting with GPU-CPU implementations for genomics workflows before expanding to broader performance portability solutions. GPU-CPU implementations offer immediate advantages for genomics workflows, providing significant performance improvements due to massive parallelism and high memory bandwidth. This approach offers immediate speedups for computationally intensive tasks and reveals optimization opportunities that might not be apparent in CPU-only implementations. The maturity of the GPU software ecosystem and growing support for GPU computing in bioinformatics tools provide a solid foundation for GPU-based genomics workflows. However, GPU implementations face challenges such as memory management, irregular memory access patterns, and load balancing. These can be addressed through unified CPU-GPU memory, efficient data streaming techniques, optimized data structures, and dynamic work distribution strategies.

While GPU-CPU implementations are a strong starting point, the diverse landscape of computing hardware in genomics, especially at the POCT, necessitates a move toward performance portability. Frameworks such as SYCL offer promising solutions, providing high-level abstractions that map efficiently to various hardware architectures. These frameworks allow code to be written once and run on multiple platforms, reducing development and maintenance efforts. Transitioning to performance-portable solutions presents its own challenges, including potential performance overhead and the need for developers to understand both genomics and performance-portable programming.

In conclusion, starting with GPU-CPU implementations provides a practical approach to accelerating genomics workflows, while the transition to performance-portable solutions represents a crucial next step. This evolution will enable the genomics community to utilize a wide range of computing architectures efficiently, accelerating genomic research and its clinical applications.

VI. Conclusion

The field of genomics stands at a critical juncture, facing unprecedented growth in data volume and analysis complexity. This review has outlined emerging trends in software and hardware for genomics, emphasizing the urgent need for innovative computational solutions. We have examined key algorithms in genomic sequencing workflows and provided insights into potential bottlenecks and optimization opportunities. GPU computing has emerged as a promising approach to accelerate these workflows, offering significant performance gains. The transition from CPU-centric to GPU-accelerated implementations is a crucial first step, but the diverse and evolving hardware landscape necessitates further evolution toward performance portability. Frameworks like SYCL offer promising solutions to achieve this goal. Advancements in edge and confidential computing, combined with ongoing efforts in data compression and novel storage solutions like DNA-based storage, will shape the future of genomics. The genomics community can unlock new possibilities in personalized medicine and disease research by embracing HPC, performance portability, and emerging technologies like ML.

[1] M. van Lanschot, L. Bosch, M. de Wit, B. Carvalho, G. Meijer, “Early detection: The impact of genomics", Virchows Archiv, vol. 471, no. 2, pp. 165–173, 2017.

[2] C. H. June, R. S. O’Connor, O. U. Kawalekar, S. Ghassemi, M. C. Milone, “Car t cell immunotherapy for human cancer,” Science, vol. 359, no. 6382, pp. 1361–1365, 2018.

[3] E. Zeggini, A. L. Gloyn, A. C. Barton, L. V. Wain, “Translational genomics and precision medicine: Moving from the lab to the clinic,” Science, vol. 365, no. 6460, pp. 1409–1413, 2019.

[4] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?," in PLoS Biology, vol. 13, no. 7, p. e1002195, Jul. 2015.

[5] “Critical and emerging technologies list update a report by the fast track action subcommittee on critical and emerging technologies of the national science and technology council,” 2022. Available: https://www.whitehouse.gov/wp-content/uploads/2022/02/02-2022-Critical-and-Emerging-Technologies-List-Update.pdf

[6] GA4GH. (n.d.). *Cram: The genomics compression standard*. The Global Alliance for Genomics and Health. unpublished.

[7] Gamaarachchi, Hasindu, Hiruna Samarakoon, Sasha P. Jenner, James M. Ferguson, Timothy G. Amos, Jillian M. Hammond, Hassaan Saadat, Martin A. Smith, Sri Parameswaran, and Ira W. Deveson. "Fast nanopore sequencing data analysis with SLOW5." *Nature biotechnology* 40, no. 7 (2022): 1026-1029.

[8] Y. Liu, H. Peng, L. Wong, and J. Li, “High-speed and high-ratio referential genome compression,” Bioinformatics, vol. 33, no. 21, pp. 3364-3372, 2017.

[9] S. Deorowicz, A. Danek, and S. Grabowski, “Genome compression: a novel approach for large collections,” Bioinformatics, vol. 29, no. 20, pp. 2572-2578, 2013.

[10] Y. Turakhia, G. Bejerano, and W. J. Dally, “Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly,” ACM SIGPLAN Notices, vol. 53, no. 2, pp. 199-213, 2018.

[11] A. Nag, C. N. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P. Gaillardon, “Gencache: Leveraging in-cache operators for efficient sequence alignment,” in Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, pp. 334-346, 2019.

[12] D. Fujiki, A. Subramaniyan, T. Zhang, Y. Zeng, R. Das, D. Blaauw, and S. Narayanasamy, “GenAx: A genome sequencing accelerator,” in 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), pp. 69-82, IEEE, 2018.

[13] T. Dunn, H. Sadasivan, J. Wadden, K. Goliya, K. Chen, D. Blaauw, R. Das, and S. Narayanasamy, “Squigglefilter: An accelerator for portable virus detection,” in MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 535-549, 2021.

[14] P. J. Shih, H. Saadat, S. Parameswaran, and H. Gamaarachchi, “Efficient real-time selective genome sequencing on resource-constrained devices,” GigaScience, vol. 12, 2023.

[15] T. Robinson, J. Harkin, and P. Shukla, “Hardware acceleration of genomics data analysis: challenges and opportunities,” Bioinformatics, vol. 37, no. 13, pp. 1785-1795, 2021.

[16] S. Walia, C. Ye, A. Bera, D. Lodhavia, and Y. Turakhia, “TALCO: Tiling Genome Sequence Alignment Using Convergence of Traceback Pointers,” in 2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA), pp. 91-107, IEEE, 2024.

[17] Y. Gu, A. Subramaniyan, T. Dunn, A. Khadem, K.-Y. Chen, S. Paul, M. Vasimuddin, et al., “GenDP: A Framework of Dynamic Programming Acceleration for Genome Sequencing Analysis,” in Proceedings of the 50th Annual International Symposium on Computer Architecture, pp. 1-15, 2023.

[18] L. López-Villellas, R. Langarita-Benítez, A. Badouh, V. Soria-Pardos, Q. Aguado-Puig, G. López-Paradís, M. Doblas, et al., “GenArchBench: A genomics benchmark suite for arm HPC processors,” Future Generation Computer Systems, vol. 157, pp. 313-329, 2024.

[19] A. Subramaniyan, Y. Gu, T. Dunn, S. Paul, M. Vasimuddin, S. Misra, D. Blaauw, S. Narayanasamy, and R. Das, “Genomicsbench: A benchmark suite for genomics,” in 2021 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 1-12, IEEE, 2021.

[20] O. Mutlu and C. Firtina, “Accelerating genome analysis via algorithm-architecture co-design,” in 2023 60th ACM/IEEE Design Automation Conference (DAC), pp. 1-4, IEEE, 2023.

[21] D. S. Cali, K. Kanellopoulos, J. Lindegger, Z. Bingöl, G. S. Kalsi, Z. Zuo, C. Firtina, M. B. Cavlak, J. Kim, N. M. Ghiasi, and G. Singh, “SeGraM: A universal hardware accelerator for genomic sequence-to-graph and sequence-to-sequence mapping,” in Proceedings of the 49th Annual International Symposium on Computer Architecture, pp. 638-655, June 2022.

[22] H. Sadasivan, D. Stiffler, A. Tirumala, J. Israeli, and S. Narayanasamy, “Accelerated dynamic time warping on GPU for selective nanopore sequencing,” bioRxiv, 2023. unpublished.

[23] J. Dong, X. Liu, H. Sadasivan, S. Sitaraman, and S. Narayanasamy, “mm2-gb: GPU Accelerated Minimap2 for Long Read DNA Mapping,” bioRxiv, 2024.

[24] H. Sadasivan, M. Maric, E. Dawson, V. Iyer, J. Israeli, and S. Narayanasamy, “Accelerating Minimap2 for accurate long read alignment on GPUs,” Journal of Biotechnology and Biomedicine, vol. 6, no. 1, p. 13, 2023.

[25] Y. Choi, H. J. Bae, A. C. Lee, H. Choi, D. Lee, T. Ryu, J. Hyun, S. Kim, H. Kim, S. H. Song, and K. Kim, “DNA micro-disks for the management of DNA-based data storage with index and write-once–read-many (WORM) memory features,” Advanced Materials, vol. 32, no. 37, p. 2001249, 2020.

[26] A. Doricchi, C. M. Platnich, A. Gimpel, F. Horn, M. Earle, G. Lanzavecchia, A. L. Cortajarena, L. M. Liz-Marzán, N. Liu, R. Heckel, R. N. Grass, R. Krahne, U. F. Keyser, and D. Garoli, “Emerging approaches to DNA data storage: challenges and prospects,” ACS Nano, vol. 16, no. 11, pp. 17552-17571, Oct. 2022

[27] S. Loose, S. Malla, and M. Stout, “Real-time selective sequencing using nanopore technology,” Nature Methods, vol. 13, no. 9, pp. 751-754, 2016.

[28] H. Sadasivan, J. Wadden, K. Goliya, P. Ranjan, R. P. Dickson, D. Blaauw, R. Das, and S. Narayanasamy, “Rapid real-time squiggle classification for read until using RawMap,” Archives of Clinical and Biomedical Research, vol. 7, no. 1, pp. 45-57, Jan. 2023

[29] S. Kovaka, Y. Fan, B. Ni, W. Timp, and M. C. Schatz, “Targeted nanopore sequencing by real-time mapping of raw electrical signal with UNCALLED,” Nature Biotechnology, vol. 39, no. 4, pp. 431-441, Apr. 2021C.

[30] Firtina, N. M. Ghiasi, J. Lindegger, G. Singh, M. B. Cavlak, H. Mao, and O. Mutlu, “RawHash: enabling fast and accurate real-time analysis of raw nanopore signals for large genomes,” Bioinformatics, vol. 39, no. Supplement_1, pp. i297-i307, 2023.

[31] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” IEEE Internet of Things Journal, vol. 3, no. 5, pp. 637-646, Oct. 2016.

[32] M. Jofre, D. Navarro-Llobet, R. Agulló, J. Puig, G. Gonzalez-Granadillo, J. Mora Zamorano, and R. Romeu, “Cybersecurity and privacy risk assessment of point-of-care systems in healthcare—a use case approach,” Applied Sciences, vol. 11, no. 15, p. 6699, Jul. 2021

[33] H. H. Ng, H. C. Ang, S. Y. Hoe, M. L. Lim, H. E. Tai, R. C. H. Soh, & C. K.-C. Syn, “Simple DNA extraction of urine samples: Effects of storage temperature and storage time,” Forensic Science International, 287, pp. 36-39, 2018.

[34] S. Chaterji, J. Koo, N. Li, F. Meyer, A. Grama, & S. Bagchi, “Federation in genomics pipelines: techniques and challenges,” Briefings in Bioinformatics, vol. 20, no. 1, pp. 235-244, 2019.

[35] H. Li, “Minimap2: pairwise alignment for nucleotide sequences,” Bioinformatics, vol. 34, no. 18, pp. 3094-3100, Sep. 2018

[36] D. E. Wood and S. L. Salzberg, “Kraken: ultrafast metagenomic sequence classification using exact alignments,” Genome Biology, vol. 15, Article number: R46, Mar. 2014

[37] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with bowtie 2,” Nature methods, vol. 9, no. 4, p. 357, 2012.

[38] H. Li, “Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM,” arXiv preprint arXiv:1303.3997, 20131. unpublished.

[39] D. Kim, L. Song, F. P. Breitwieser, and S. L. Salzberg, “Centrifuge: rapid and sensitive classification of metagenomic sequences,” Genome research, vol. 26, no. 12, pp. 1721–1729, 2016.

[40] P. E. C. Compeau, P. A. Pevzner, and G. Tesler, “How to apply de Bruijn graphs to genome assembly,” Nature Biotechnology, vol. 29, no. 11, pp. 987-991, Nov. 2011.

[41] “Genome analysis toolkit: Variant discovery in high-throughput sequencing data,” https://software.broadinstitute.org/gatk/.

[42] “Germline short variant discovery (snps + indels). best practices workflow,”https://software.broadinstitute.org/gatk/best-practices/workflow?id=11145.

[43] M. Kolmogorov, J. Yuan, Y. Lin, and P. A. Pevzner, “Assembly of long, error-prone reads using repeat graphs,” Nature Biotechnology, vol. 37, no. 5, pp. 540-546, 2019.

[44] A. Rimmer, H. Phan, I. Mathieson, Z. Iqbal, S. R. Twigg, A. O.Wilkie, G. McVean, and G. Lunter, “Integrating mapping-, assembly-and haplotype-based approaches for calling variants in clinical sequencing applications,” Nature genetics, vol. 46, no. 8, pp. 912–918, 2014

[45] S. Koren, B. P. Walenz, K. Berlin, J. R. Miller, N. H. Bergman, and A. M. Phillippy, “Canu: scalable and accurate long-read assembly via adaptive k-mer weighting and repeat separation,” Genome Research, vol. 27, no. 5, pp. 722-736, 2017.

[46] R. Vaser, I. Sović, N. Nagarajan, and M. Šikić, “Fast and accurate de novo genome assembly from long uncorrected reads,” Genome Research, vol. 27, no. 5, pp. 737-746, 2017.

[47] S. Kalikar, C. Jain, M. Vasimuddin, and S. Misra, “Accelerating minimap2 for long-read sequencing applications on modern CPUs,” Nature Computational Science, vol. 2, no. 2, pp. 78-83, 2022.

[48] M. H.-Y. Fritz, R. Leinonen, G. Cochrane, and E. Birney, “Efficient storage of high throughput DNA sequencing data using reference-based compression,” Genome Research, vol. 21, no. 5, pp. 734-740, 2011.

[49] A. Chacón, S. Marco-Sola, A. Espinosa, P. Ribeca, and J. C. Moure, “Boosting the FM-index on the GPU: Effective techniques to mitigate random memory access,” IEEE/ACM Transactions on Computational Biology and Bioinformatics, vol. 12, no. 5, pp. 1048-1059, 2014.

[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,” Journal of Molecular Biology, vol. 48, no. 3, pp. 443-453, 1970.

[51] T. F. Smith and M. S. Waterman, “Identification of common molecular subsequences,” Journal of Molecular Biology, vol. 147, no. 1, pp. 195-197, 1981.

[52] O. Gotoh, “An improved algorithm for matching biological sequences,” Journal of Molecular Biology, vol. 162, no. 3, pp. 705-708, 1982.

[53] N. J. Loman, J. Quick, and J. T. Simpson, “A complete bacterial genome assembled de novo using only nanopore sequencing data,” Nature Methods, vol. 12, no. 8, pp. 733-735, 2015.

[54] C. Lee, C. Grasso, and M. F. Sharlow, “Multiple sequence alignment using partial order graphs,” Bioinformatics, vol. 18, no. 3, pp. 452-464, 2002.

[55] G. Myers, “A fast bit-vector algorithm for approximate string matching based on dynamic programming,” Journal of the ACM (JACM), vol. 46, no. 3, pp. 395-415, 1999.

[56] S. Marco-Sola, J. C. Moure, M. Moreto, and A. Espinosa, “Fast gap-affine pairwise alignment using the wavefront algorithm,” Bioinformatics, vol. 37, no. 4, pp. 456-463, 2021.

[57] H. Gamaarachchi, C. W. Lam, G. Jayatilaka, H. Samarakoon, J. T. Simpson, M. A. Smith, and S. Parameswaran, “GPU accelerated adaptive

banded event alignment for rapid comparative nanopore signal analysis," BMC Bioinformatics, vol. 21, pp. 1-13, 2020.

[58] H. Suzuki and M. Kasahara, "Introducing difference recurrence relations for faster semi-global alignment of long sequences," BMC Bioinformatics, vol. 19, pp. 33-47, 2018.

[59] B. Schmidt and C. Hundt, "cuDTW++: ultra-fast dynamic time warping on CUDA-enabled GPUs," in Euro-Par 2020: Parallel Processing: 26th International Conference on Parallel and Distributed Computing, Warsaw, Poland, August 24–28, 2020, Proceedings 26, pp. 597-612. Springer International Publishing, 2020.

[60] L. Guo, J. Lau, Z. Ruan, P. Wei, and J. Cong, "Hardware acceleration of long read pairwise overlapping in genome sequencing: A race between fpga and gpu," in 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), pp. 127-135. IEEE, 2019.

[61] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," Advances in Neural Information Processing Systems 30, 2017.

[62] Oxford Nanopore Technologies, "Dorado," 2023. [Online]. Available: https://github.com/nanoporetech/dorado. [Accessed: 13-Jun-2023].

[63] R. Luo, C.-L. Wong, Y.-S. Wong, C.-I. Tang, C.-M. Liu, C.-M. Leung, and T.-W. Lam, "Exploring the limit of using a deep neural network on pileup data for germline variant calling," Nature Machine Intelligence, vol. 2, no. 4, pp. 220-227, 2020.

[64] Z. Zheng, S. Li, J. Su, A. W.-S. Leung, T.-W. Lam, and R. Luo, "Symphonizing pileup and full-alignment for deep learning-based long-read variant calling," Nature Computational Science, vol. 2, no. 12, pp. 797-803, 2022.

[65] R. Poplin, P.-C. Chang, D. Alexander, S. Schwartz, T. Colthurst, A. Ku, D. Newburger, et al., "A universal SNP and small-indel variant caller using deep neural networks," Nature Biotechnology, vol. 36, no. 10, pp. 983-987, 2018.

[66] Medaka, (2023). [Online]. Available: https://github.com/nanoporetech/medaka. (Accessed: 1-Jul-2024).

[67] R. R. Wick, L. M. Judd, and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," Genome Biology, vol. 20, pp. 1-10, 2019.

[68] "ROCm/composable_kernel: Composable Kernel: Performance Portable Programming Model for Machine Learning Tensor Operators," GitHub, n.d. [Online]. Available: https://github.com/ROCm/composable_kernel. [Accessed: 1-Jul-2024].

[69] Nvidia. (n.d.). GitHub - NVIDIA/cutlass: CUDA Templates for Linear Algebra Subroutines. GitHub. https://github.com/NVIDIA/cutlass/

[70] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "vllm: Easy, fast, and cheap llm serving with pagedattention," 2023. [Online]. Available: https://vllm.ai/. [Accessed: 1-Jul.- 2024].

[71] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai, "Gqa: Training generalized multi-query transformer models from multi-head checkpoints," arXiv preprint arXiv:2305.13245, 2023. unpublished.

[72] J. K. Bonfield, "CRAM 3.1: advances in the CRAM file format," Bioinformatics, vol. 38, no. 6, pp. 1497-1503, 2022.

[73] M. Costanzo, E. Rucci, C. García-Sanchez, M. Naiouf, and M. Prieto-Matías, "Assessing opportunities of SYCL for biological sequence alignment on GPU-based systems," The Journal of Supercomputing, pp. 1-24, 2024.

[74] A. Maxmen, "One million coronavirus sequences: popular genome site hits mega milestone," Nature, vol. 593, no. 7857, pp. 21-21, 2021.

[75] J. November, "More than Moore's mores: computers, genomics, and the embrace of innovation," Journal of the History of Biology, vol. 51, no. 4, pp. 807-840, 2018.

[76] J. Rosenblum, J. Dong, and S. Narayanasamy, "SECRET-GWAS: Confidential Computing for Population-Scale GWAS," bioRxiv, 2024. unpublished.

[77] Organick, L., Ang, S., Chen, YJ. et al. Random access in large-scale DNA data storage. Nat Biotechnol 36, 242–248 (2018).

[78] "Google Cloud Confidential Computing Powered by AMD," AMD, [Online]. Available: https://www.amd.com/en/products/processors/server/epyc/google-cloud/confidential-computing.html. [Accessed: 1-Jul-2024].

[79] Occupational Safety and Health Administration (OSHA), Laboratory Safety Guidance, [Online], Available: https://www.osha.gov/sites/default/files/publications/OSHA3404laboratory-safety-guidance.pdf [Accessed: 1-Jul-2024]

[80] M. Alser, J. Lindegger, C. Firtina, N. Almadhoun, H. Mao, G. Singh, J. Gomez-Luna, and O. Mutlu, "From molecules to genomic variations: Accelerating genome analysis via intelligent algorithms and architectures," Computational and Structural Biotechnology Journal, vol. 20, pp. 4579-4599, 2022.

[81] "Benchmarking the Oxford Nanopore Technologies basecallers on AWS," Amazon Web Services, 24-Nov-2023. [Online]. Available: https://aws.amazon.com/blogs/hpc/benchmarking-the-oxford-nanopore-technologies-basecallers-on-aws. [Accessed: 1-Jul.-2024].

[82] T. J. Ham et al., "Genesis: A Hardware Acceleration Framework for Genomic Data Analysis," 2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA), Valencia, Spain, 2020, pp. 254-267

[83] M. Doblas, O. Lostes-Cazorla, Q. Aguado-Puig, N. Cebry, P. Fontova-Musté, C. F. Batten, S. Marco-Sola, and M. Moretó, "Gmx: Instruction set extensions for fast, scalable, and efficient genome sequence alignment," in Proceedings of the 56th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 1466-1480, 2023.

[84] G. Singh, M. Alser, K. Denolf, C. Firtina, A. Khodamoradi, M. B. Cavlak, H. Corporaal, and O. Mutlu, "RUBICON: a framework for designing efficient deep learning-based genomic basecallers," Genome Biology, vol. 25, no. 1, p. 49, 2024.

[85] B. D. Rouhani, R. Zhao, A. More, M. Hall, A. Khodamoradi, S. Deng, D. Choudhary, et al., "Microscaling data formats for deep learning," arXiv preprint arXiv:2310.10537, 2023. unpublished.

[86] NVIDIA Parabricks v4.3.0 - NVIDIA Docs," NVIDIA Docs, [Online]. Available: https://docs.nvidia.com/clara/parabricks/4.3.0/index.html. [Accessed: 1-Jul.-2024]

[87] N. Kono and K. Arakawa, "Nanopore sequencing: Review of potential applications in functional genomics," Development, Growth & Differentiation, vol. 61, no. 5, pp. 316–326, 2019. [Online]. Available: https://doi.org/10.1111/dgd.12608

[88] J. Hench, C. Hultschig, J. Brugger, L. Mariani, R. Guzman, J. Soleman, S. Leu et al., "EpiDiP/NanoDiP: a versatile unsupervised machine learning edge computing platform for epigenomic tumour diagnostics," Acta Neuropathologica Communications, vol. 12, no. 1, p. 51, 2024.

[89] E. B. Hodcroft, N. De Maio, R. Lanfear, D. R. MacCannell, B. Q. Minh, H. A. Schmidt, A. Stamatakis, N. Goldman, and C. Dessimoz, "Want to track pandemic variants faster? Fix the bioinformatics bottleneck," Nature, vol. 591, no. 7848, pp. 30-33, 2021.

[90] S. Deorowicz, A. Danek, and H. Li, "AGC: compact representation of assembled genomes with fast queries and updates," Bioinformatics, vol. 39, no. 3, 2023, Art. no. btad097.

[91] K. Břinda, L. Lima, S. Pignotti, N. Quinones-Olvera, K. Salikhov, R. Chikhi, G. Kucherov, Z. Iqbal, and M. Baym, "Efficient and Robust Search of Microbial Genomes via Phylogenetic Compression," bioRxiv, 2023. unpublished.

[92] S. R. Eddy, "What is dynamic programming?," in Nature Biotechnology, vol. 22, no. 7, pp. 909-910, 2004.

[93] E. Nguyen, M. Poli, M. G. Durrant, A. W. Thomas, B. Kang, J. Sullivan, M. Y. Ng, A. Lewis, A. Patel, and A. Lou, "Sequence modeling and design from molecular to genome scale with evo," bioRxiv, 2024. unpublished.

[94] E. Nguyen, M. Poli, M. Faizi, A. Thomas, M. Wornow, C. Birch-Sykes, S. Massaroli, et al., "Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution," Advances in Neural Information Processing Systems 36, 2024.

[95] W. J. Dally, Y. Turakhia, and S. Han, "Domain-specific hardware accelerators," Communications of the ACM, vol. 63, no. 7, pp. 48-57, 2020.

[96] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen et al., "Pytorch: An imperative style, high-performance deep learning library," Advances in Neural Information Processing Systems, vol. 32, 2019.

[97] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265-283, 2016.

[98] C.R. Trott, D. Lebrun-Grandie, D. Arndt, et al., "Kokkos 3 Programming Model Extensions for the Exascale Era" IEEE Transactions on Parallel and Distributed Systems, vol. 33, no. 4, pp. 805-817, 1 April 2022, doi: 10.1109/TPDS.2021.3097283.

[99] J. Ragan-Kelley, C. Barnes, A. Adams, S. Paris, F. Durand, and S. Amarasinghe, "Halide: a language and compiler for optimizing parallelism, locality, and recomputation in image processing pipelines," ACM SIGPLAN Notices, vol. 48, no. 6, pp. 519-530, 2013.

[100] Fahim, Farah, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran, Luca P. Carloni et al. "hls4ml: An open-source codesign workflow to empower scientific low-power machine learning devices." arXiv preprint arXiv:2103.05579 (2021).

[101] S. Walia, H. Motwani, K. Smith, R. Corbett-Detig, and Y. Turakhia, "Compressive Pangenomics Using Mutation-Annotated Networks," bioRxiv, 2024. unpublished.

[102] L. Ceze, J. Nivala, and K. Strauss, "Molecular digital data storage using DNA," Nature Reviews Genetics, vol. 20, no. 8, pp. 456-466, 2019.

[103] N. M. Ghiasi, J. Park, H. Mustafa, J. Kim, A. Olgun, A. Gollwitzer, D. S. Cali et al., "GenStore: A High-Performance and Energy-Efficient In-Storage Computing System for Genome Sequence Analysis," arXiv preprint arXiv:2202.10400, 2022. unpublished.

[104] M. Alser, Z. Bingöl, D. S. Cali, J. Kim, S. Ghose, C. Alkan, and O. Mutlu, "Accelerating genome analysis: A primer on an ongoing journey," IEEE Micro, vol. 40, no. 5, pp. 65-75, 2020.

[105] Z. Bingöl, M. Alser, O. Mutlu, O. Ozturk, and C. Alkan, “GateKeeper-GPU: Fast and accurate pre-alignment filtering in short read mapping,” IEEE Transactions on Computers, 2024.

[106] G. Singh, M. Alser, D. S. Cali, D. Diamantopoulos, J. Gómez-Luna, H. Corporaal, and O. Mutlu, “FPGA-based near-memory acceleration of modern data-intensive applications,” IEEE Micro, vol. 41, no. 4, pp. 39-48, 2021.

[107] M. Alser, H. Hassan, A. Kumar, O. Mutlu, and C. Alkan, “Shouji: a fast and efficient pre-alignment filter for sequence alignment,” Bioinformatics, vol. 35, no. 21, pp. 4255-4263, 2019.

[108] M. Alser, T. Shahroodi, J. Gómez-Luna, C. Alkan, and O. Mutlu, “SneakySnake: a fast and accurate universal genome pre-alignment filter for CPUs, GPUs and FPGAs,” Bioinformatics, vol. 36, no. 22-23, pp. 5282-5290, 2020.

[109] J. R. Pohlhaus and R. M. Cook-Deegan, “Genomics research: world survey of public funding,” BMC Genomics, vol. 9, pp. 1-18, 2008.

[110] A. Smith et al., "11.1 AMD InstinctTM MI300 Series Modular Chiplet Package–HPC and AI Accelerator for Exa-Class Systems," in 2024 IEEE International Solid-State Circuits Conference (ISSCC), vol. 67, IEEE, 2024, pp. 490-492.

[111] D. Fujiki et al., "GenAx: A genome sequencing accelerator," in 2018 ACM/IEEE 45th Annual International

[112] A. Subramaniyan et al., "Accelerated seeding for genome sequence alignment with enumerated radix trees," in 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021, pp. 388-401. Symposium on Computer Architecture (ISCA), 2018, pp. 69-82.

[113] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000 x acceleration on long read assembly," ACM SIGPLAN Notices, vol. 53, no. 2, pp. 199-213, 2018.

[114] M. F. Chang et al., "The smem seeding acceleration for dna sequence alignment," in 2016 IEEE 24th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), 2016, pp. 32-39.

[115] W. Huangfu, S. Li, X. Hu, and Y. Xie, "RADAR: A 3D-ReRAM based DNA alignment accelerator architecture," in Proceedings of the 55th Annual Design Automation Conference, 2018, pp. 1-6.

[116] R. Kaplan, L. Yavits, R. Ginosar, and U. Weiser, "A resistive CAM processing-in-storage architecture for DNA sequence alignment," IEEE Micro, vol. 37, no. 4, pp. 20-28, 2017.

[117] V. Zois, D. Gupta, V. J. Tsotras, W. A. Najjar, and J. F. Roy, "Massively parallel skyline computation for processing-in-memory architectures," in Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques, 2018, pp. 1-12.

[118] W. Huangfu et al., "Medal: Scalable dimm based near data processing accelerator for dna seeding algorithm," in Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture, 2019, pp. 587-599.