bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

RawHash: Enabling Fast and Accurate Real-Time Analysis of Raw Nanopore Signals for Large Genomes

Can Firtina[1] Nika Mansouri Ghiasi[1] Joel Lindegger[1] Gagandeep Singh[1]
Meryem Banu Cavlak[1] Haiyu Mao[1] Onur Mutlu[1]
[1]*ETH Zurich*

Nanopore sequencers generate electrical raw signals in real-time while sequencing long genomic strands. These raw signals can be analyzed as they are generated, providing an opportunity for real-time genome analysis. An important feature of nanopore sequencing, Read Until, can eject strands from sequencers without fully sequencing them, which provides opportunities to computationally reduce the sequencing time and cost. However, existing works utilizing Read Until either 1) require powerful computational resources that may not be available for portable sequencers or 2) lack scalability for large genomes, rendering them inaccurate or ineffective.

We propose RawHash, the first mechanism that can accurately and efficiently perform real-time analysis of nanopore raw signals for large genomes using a hash-based similarity search. To enable this, RawHash ensures the signals corresponding to the same DNA content lead to the same hash value, regardless of the slight variations in these signals. RawHash achieves an accurate hash-based similarity search via an effective quantization of the raw signals such that signals corresponding to the same DNA content have the same quantized value and, subsequently, the same hash value.

We evaluate RawHash on three applications: 1) read mapping, 2) relative abundance estimation, and 3) contamination analysis. Our evaluations show that RawHash is the only tool that can provide high accuracy and high throughput for analyzing large genomes in real-time. When compared to the state-of-the-art techniques, UNCALLED and Sigmap, RawHash provides 1) 25.8× *and* 3.4× *better average throughput and 2) an average speedup of* 32.1× *and* 2.1× *in the mapping time, respectively. Source code is available at https://github.com/CMU-SAFARI/RawHash.*

1. Introduction

High-throughput sequencing (HTS) devices can generate a large amount of genomic data at a relatively low cost [1]. HTS can be used to analyze a wide range of samples, from small amounts of DNA or RNA to entire genomes. Oxford Nanopore Technologies (ONT) is one of the most widely-used HTS technologies that can sequence long genomic regions, called *reads*, with up to a few million bases [2]. ONT devices use the nanopore sequencing technique, which involves passing a single DNA or RNA strand through a tiny pore, *nanopore*, at an average speed of 450 bases per second [3] and measuring the electrical current as the strand passes through. Nanopore sequencing enables two key features. First, nanopores provide the electrical raw signals in *real-time* as the DNA strand passes through a nanopore. Second, nanopore sequencing provides a functionality, known as *Read Until* [4], that can partially sequence DNA strands without fully sequencing them. These two features of nanopores provide opportunities for 1) real-time genome analysis and 2) significantly reducing sequencing time and cost.

Real-time analysis of nanopore raw signals using Read Until can reduce the sequencing time and cost per read by terminating the sequencing of a read whenever sequencing the full read is not necessary. The freed-up nanopore can then be used to sequence a different read. A purely computational mechanism can send a signal to *eject* a read from a nanopore if the partial sequencing of a read meets certain conditions for particular genome analysis, such as 1) reaching a desired coverage for a species in a sample [5] or 2) identifying that a read does not originate from a certain genome of interest (i.e., a target region) [3, 6] and hence, does not need to be fully sequenced. By terminating the sequencing of reads that do not correspond to the target region, the sequencer can spend time and resources on higher coverage sequencing of the reads that correspond to the target. This process is referred to as *targeted sequencing*. By providing high coverage at target regions and avoiding unessential sequencing of reads outside those regions, this approach can improve the quality of sequencing and the downstream analysis utilizing the obtained data.

To effectively utilize the Read Until technique in nanopore sequencing, it is crucial to have computational methods that can accurately analyze the raw output signals from nanopores in real-time. These methods must provide 1) low latency and 2) throughput matching or exceeding that of the sequencer [3, 6, 7]. Several works propose methods for real-time analysis of raw nanopore signals with Read Until [3, 5–11]. However, these works have three key limitations. First, most techniques require powerful computational resources, such as GPUs [5, 9], or specialized hardware [7, 10] due to the use of computationally-intensive algorithms. This can make real-time genome analysis challenging for portable and low-cost nanopore-based sequencers, such as the ONT Flongle or MinION, which are not typically equipped with such resources. Therefore these techniques cannot be used in resource-constrained environments. Second, the sheer size of genomic data at the scale of large genomes (e.g., human genome) makes it challenging to process the data in real-time. This is because such large genomes require efficient and accurate similarity identification across a large number of regions. This renders many current methods [3, 6] inaccurate or useless for large genomes as they cannot either provide accurate results or match the throughput of nanopores for these genomes. Third, machine learning models used in past works [5, 8, 9] to analyze raw nanopore signals often require retraining or reconfiguring the model to improve accuracy for a certain experiment [11, 12], which can be a barrier to flexibly and easily performing real-time analysis without retraining or reconfiguring these models. To our knowledge, there is no work that can efficiently and accurately perform real-time analysis of raw nanopore signals on a large scale (e.g., whole-genome analysis for human) without requiring powerful computational resources, which can easily and flexibly be applied to a wide range of applications that could benefit from real-time nanopore raw signal analysis.

Our **goal** is to enable efficient and accurate real-time genome analysis for large genomes. To this end, we propose *RawHash*, the *first* mechanism that can efficiently and accurately perform

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

real-time analysis of raw nanopore signals for large genomes in resource-contained environments. Unlike all the past works, RawHash is the only mechanism that can efficiently scale to large genomes and perform accurate real-time genomic analysis without requiring dedicated GPU support. Our **key idea** is to encode regions of the raw nanopore signal into hash values such that similar signal regions can efficiently be identified by matching their hash values, facilitating efficient similarity identification between signals. However, enabling accurate hashing-based similarity identification in the raw signal domain is challenging because raw signals corresponding to the same DNA content are unlikely to have exactly the same signal amplitudes. This is because the raw signals generated by nanopores can vary each time the same DNA fragment is sequenced due to several factors impacting nanopores during sequencing, such as variations in the properties of the nanopores or the conditions in which the sequencing is performed [13]. Although the similarity identification of raw signals is possible via calculating the Euclidean distance between a sequence of signals in a multi-dimensional space [6], such an approach can become impractical when dealing with larger sequences as the number of dimensions increases with the length of the sequences. This increase in dimensionality can lead to computational complexity and the curse of dimensionality, making it expensive and impractical.

To address these challenges, RawHash provides three **key mechanisms** for efficient signal encoding and similarity identification. First, RawHash encodes signal values that have a wider range of values into a smaller set of values using a quantization technique, such that signal values within a certain range are assigned to the same encoded value. This helps to alleviate the probability of having varying signal values for the same DNA content and enables RawHash to directly match these values using a hashing technique. Second, RawHash concatenates the quantized values of *multiple* consecutive signals and generates a single hash value for them. The hashing mechanism enables RawHash to efficiently identify similar signal regions of these consecutive signal values by directly matching their corresponding hash values. Representing many consecutive signals with a single hash value increases the size of the regions examined during similarity identification without suffering from the curse of dimensionality. Using larger regions can substantially reduce the number of possible matching regions that need to be examined. RawHash is the *first* work that can accurately use hash values in the raw signal domain, which enables using efficient data structures commonly-used used in the sequence domain (e.g., hash tables in minimap2 [14]). Third, RawHash uses an existing algorithm, known as chaining [14, 15], to find the colinear matches of hash values between signals to identify similar signal regions. These efficient and accurate mechanisms enable RawHash to perform real-time genome analysis for large genomes.

While our proposed three key mechanisms have the potential to be used for various purposes in raw signal similarity identification, we design RawHash as a tool for mapping nanopore raw signals to their corresponding reference genomes in real-time. RawHash operates the mapping in two steps 1) *indexing* and 2) *mapping*. First, in the indexing step, RawHash 1) converts the reference genome sequence into *expected* signal values by simulating the expected behavior of nanopores based on a previously-known model, 2) generates the hash values from these signals, and 3) stores the hash values in a hash table for efficient matching. Second, in the mapping step, RawHash 1) generates the hash values from the raw signals in a streaming fashion, 2) queries the hash table from the indexing step with these hash values to find the matching regions in the reference genome with the same hash value, and 3) performs chaining to find the similar region between the reference genome and the raw signal of a read.

RawHash can utilize the Read Until functionality to reduce the sequencing time and cost in two ways. First, to avoid redundant sequencing and processing of each read, RawHash can use Read Until to eject a read before it is fully sequenced if RawHash identifies that the sequenced portion of the read can already be mapped to a reference genome. Second, to perform a cost- and time-efficient relative abundance estimation, RawHash utilizes Read Until to *fully* stop the *entire* sequencing of all subsequent reads after sequencing a certain amount of reads that is sufficient to make an accurate relative abundance estimation. We refer to such usage of Read Until during abundance estimation as *Sequence Until*. Avoiding the redundant sequencing of further reads that are unlikely to substantially change the relative abundance estimation has the potential to significantly reduce the sequencing time and cost. To utilize Sequence Until, RawHash integrates a confidence calculation mechanism that evaluates the relative abundance estimations in real-time and fully stops the entire sequencing run if using more reads does not change its estimation. To stop the entire sequencing run for further reads, Read Until can continuously be used for a nanopore to instantly eject the reads from the pore. We find that Sequence Until can be applied to other mechanisms (e.g., UNCALLED) that can perform real-time relative abundance estimations. Prior work [5] proposes a technique to terminate the sequencing process when species in the sample reach a certain coverage depth. The key difference of Sequence Until is that it reduces the cost of sequencing for relative abundance estimation and is based on our adaptive, accurate, and low-cost confidence calculation during real-time abundance estimation.

We evaluate RawHash on three important applications that can benefit from real-time genome analysis: 1) read mapping, 2) relative abundance estimation, and 3) contamination analysis. We compare RawHash with the state-of-the-art approaches, UNCALLED and Sigmap, which can be used with nanopore sequencers that may not be equipped with GPUs, such as the MinION devices. We evaluate RawHash, UNCALLED, and Sigmap in terms of their performance, accuracy, and their estimated benefits in reducing the sequencing time and cost.

This paper provides the following **key contributions** and **major results**:

- We propose RawHash, the *first* mechanism that can efficiently and accurately find the similarities between raw nanopore signals and a reference genome for large genomes without requiring powerful computational resources such as GPUs.
- We extensively evaluate RawHash by comparing it with state-of-the-art approaches, UNCALLED and Sigmap, on various datasets ranging from small genomes (i.e., genomes with up to 100 million bases) to large genomes (e.g., human genome). Our results show that RawHash provides 1) comparable accuracy to UNCALLED and Sigmap for small genomes and 2) significantly better accuracy for large genomes than UNCALLED and Sigmap.

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

- We show that Sigmap cannot perform real-time genome analysis for large genomes as it cannot match the throughput of nanopores.
- We provide the open source implementation of RawHash and the complete set of scripts to reproduce the results shown in this paper at https://github.com/CMU-SAFARI/RawHash.

2. Methods

We propose **RawHash**, a mechanism that can efficiently and accurately identify similarities between raw nanopore signals of a read and a large reference genome in real-time (i.e., while the read is sequenced). The raw nanopore signal of each read is a series of electrical current measurements as a strand of DNA passes through a nanopore. The reference genome is a set of strings over the alphabet `A,C,G,T`. RawHash provides the mechanisms for generating hash values from both a raw nanopore signal and a reference genome such that similar regions between the two can be efficiently and accurately found by matching their hash values.

2.1. Overview

Figure 1 shows the overview of how RawHash identifies similarities between raw nanopore signals of a read and a reference genome in four steps. First, RawHash pre-processes both 1) the raw nanopore signal and 2) the reference genome into values that are comparable to each other. For raw signals, RawHash segments the raw signal into non-overlapping regions such that each region is expected to contain a certain amount of signal values that are generated from reading a fixed number k of DNA bases. Each such region is called an *event* [13]. Each event is usually represented as a floating point value derived from the signal values in the segment. For the reference genome, RawHash translates each substring of length k (called a *k-mer*) into their *expected* event values based on the nanopore model.

The event values from the reference genome are not directly comparable to the event values from raw nanopore signals due to variability in the current measurements in nanopores generating slightly different event values for the same k-mer [13]. To generate the same values from slightly different events that may contain the same k-mer information, the second step of RawHash *quantizes* the event values from a larger set of values into a smaller set. The quantization technique ensures that the event values within a certain range are likely to be assigned to the same quantized value such that the effect of signal variation is alleviated, i.e., the same k-mer is likely assigned the same quantized value.

Due to the nature of nanopores, each event usually represents a very small k-mer of length around k=6 bases, depending on the nanopore model [6]. Such a short k-mer is likely to exist in a large number of locations in the reference genome, making it challenging to efficiently identify the correct one. To make the events more unique (i.e., such that they exist only in a small number of locations in the reference genome), the third step of RawHash combines multiple consecutive quantized events into a single hash value. These hash values can then be used to efficiently identify similar regions between raw signals and the reference genome by matching the hash values generated from their events using efficient data structures such as hash tables.

Fourth, to map a raw nanopore signal of a read to a reference genome, RawHash uses a chaining algorithm [6, 14, 15] that find colinear matching hash values generated from regions that are close to each other both in the reference genome and the raw nanopore signal.

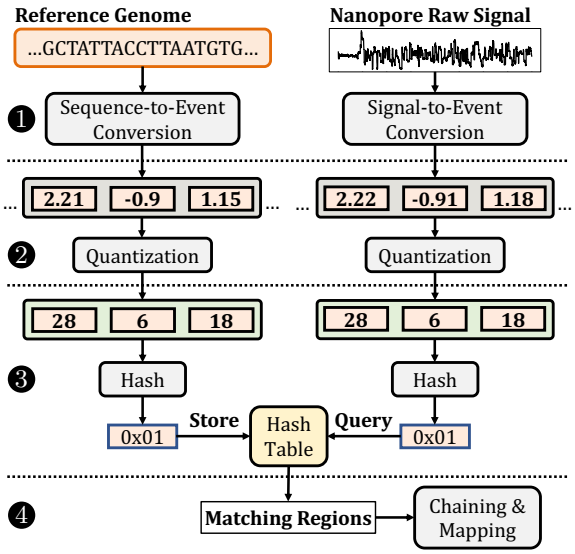

Figure 1: Overview of RawHash.

2.2. Event Generation

Our goal is to translate a reference genome sequence and a raw nanopore signal into comparable values. To this end, RawHash converts 1) each k-mer of the reference genome and 2) each segmented region of the raw signal into its corresponding event.

Sequence-to-Event Conversion. To convert a reference genome sequence into a form that can be compared with raw nanopore signals, RawHash converts the reference genome sequence into event values in three steps, as shown in Figure 2.

First, RawHash extracts all k-mers of length k from the reference genome sequence, where k depends on the nanopore. The *k-mer model* of a nanopore[1] includes the information about the *expected* k-mer length of an event and the *expected* average event value for each k-mer based on certain variables affecting the signal outcome of the nanopore's current measurements [16].

Second, RawHash queries the k-mer model for each k-mer of the reference genome to convert k-mers into their expected event values. Although the k-mer model of a nanopore provides an extensive set of information for each possible k-mer, RawHash uses only the mean values of events that provide an average value for the signals in the same event since these mean values provide a sufficient level of meaningful information for comparison with the raw nanopore signals.

Third, RawHash normalizes the event values from the same reference genome sequence (e.g., entire chromosome sequence or a contig) by calculating the standard scores (i.e., z-scores) of these events. RawHash uses these normalized values as event values since the same normalization step is taken for raw signals to avoid certain variables that may affect the range of raw signal amplitudes during sequencing [3, 6].

Signal-To-Event Conversion. Our goal is to accurately convert the series of raw nanopore signals into a set of values where each

[1]For many nanopore models, ONT provides the k-mer model [16]. For some recent nanopore models, such as the R10 devices, ONT has not provided k-mer models. In such cases, they can still be generated [17, 18]).

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

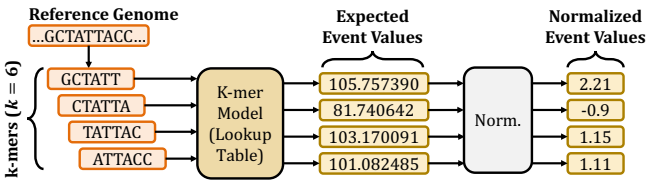

Figure 2: Converting sequences to event values based on the k-mer model of a nanopore.

value corresponds to certain DNA sequences of fixed length k, k-mers, and consecutive values differ by one base. To achieve this, RawHash converts the raw signals into their corresponding values in three steps, as shown in Figure 3. First, to accurately identify the distinct regions in the raw signal that correspond to a certain k-mer from DNA, RawHash performs a segmentation step as described in a basecalling tool, Scrappie [19], and used by earlier works UNCALLED and Sigmap. The segmentation step aims to eliminate the factors that affect the speed of the DNA molecules passing through a nanopore, as the speed affects the number of signal measurements taken for a certain amount of bases in DNA. To perform the segmentation step, RawHash identifies the boundaries in the signal where the signal value changes significantly compared to the certain amount of previously measured signal values, which indicates a base change in the nanopore. Such boundaries are computed using a statistical test, known as *Welch's t-test* [20], over a rolling window of consecutive signals. RawHash performs this t-test for multiple windows of different lengths to avoid the variables that cause a change in the number of current measurements due to the varying speed of DNA through a nanopore, known as *skip* and *stay* errors [13]. Signals that fall within the same segment (i.e., between the same measured boundaries) are usually called *events* since each event contains the signals from a reading of a fixed amount of DNA bases, k-mers.

Second, since the number of signals that each event includes is not constant across different events due to the stay and skip errors, RawHash generates a single value for each event to quickly avoid these potential errors and other factors that cause variations from reading the same amount of DNA bases. To this end, RawHash measures the mean value of the signals that fall within the same segment and uses this mean value for an event.

Third, since the amplitudes of the signal measurements may significantly vary when reading k-mers at different times, RawHash normalizes the mean event values based on the mean event values generated from the nanopore within the same certain time interval in a streaming fashion. Although this time interval parameter can be modified in our tool, the default configuration of RawHash processes the events of signals generated by the nanopore within one second. For normalization, RawHash uses the same z-score calculation that it uses for normalizing the event values generated from reference sequences as described earlier. RawHash uses these normalized values as event values when comparing with the event values from reference sequences.

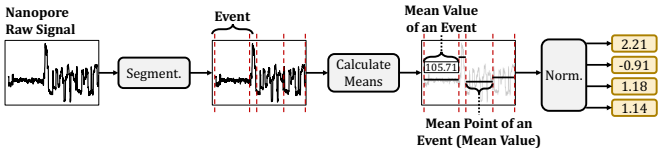

Figure 3: Detecting events from raw signals.

2.3. Quantization of Events

Our goal is to avoid the effects of generating different event values when reading the same k-mer content from nanopores so that we can identify k-mer matches by directly matching events. Although the segmentation and normalization steps explained in Section 2.2 can avoid the potential sequencing errors, such as stay and skip errors and significant changes in the current readings at different times, these approaches still do not guarantee to generate *exactly* the same event values when reading the *same* k-mer content. This is because *slight* changes in the normalized event values may occur when reading the same DNA content due to the high sensitivity and stochasticity of nanopores [13]. Thus, it is challenging to generate the same event value for the same k-mer content after the segmentation and normalization steps. Since these event values generated from reading the same k-mer content are expected to be close to each [6], we propose a quantization mechanism that encodes event values so that events with close mean values can have the same quantized value in two steps as shown in Figure 4.

First, to increase the probability of assigning the same value for similar event values, RawHash trims the least significant fractional part of mean values by using only the most significant Q bits of these mean event values from their binary format, which we represent as $E[1,Q]$ for simplicity where E is the event value and $E[1,Q]$ gives the most significant Q bits of E. We assume that the mean event values are represented by the standard single-precision floating-point format with the sign, exponent, and fraction bits. This enables RawHash to reduce the wide range of floating-point values into a smaller range *without* significantly losing from the accuracy such that event values closer to each other can be represented by the same value in the smaller range of values. We can perform this trimming technique without significant sensitivity loss because we observe that these normalized event values mostly use at most six digits from the fractional part of their values, leaving a large number of fractional bits useless.

Second, to avoid using redundant bits that may carry little or no information in the most significant Q bits of an event value, RawHash prunes p bits after the most significant two bits of $E[1,Q]$ such that $2+p<Q$ and the resulting *quantized value* is $E[1,2]E[3+p,Q]$. For simplicity, we show the quantized value of E as $E_{Q,p}$. By ignoring these p bits, we effectively pack Q bits into $Q-p$ bits without losing significant information from event values. We can perform such a pruning operation because we observe that the normalized event values are usually between in the range $[-3,3]$ such that these p bits provide little information in distinguishing different event values due to the small range of values. We note that these Q and p values are parameters to RawHash and can empirically be adjusted based on the required sensitivity and quantization efficiency. This quantization technique enables RawHash to assign the same quantized values for a pair of *close* event values, E and F, that may be generated from reading the same k-mer such that $E_{Q,p}=F_{Q,p}$ where $|E-F|<\varepsilon$ and ε is small enough for two events to represent the same k-mer content. RawHash always uses the most significant two bits as these two bits consistently carry the most significant information of the normalized event values, including the sign bit.

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

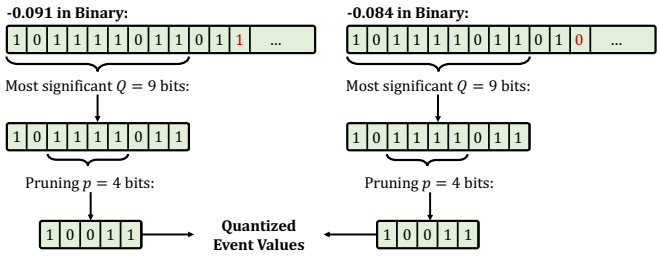

Figure 4: Quantization of two event values.

2.4. Generating the Hash Values

Our goal is to generate values for *large* regions of raw nanopore signals and reference sequences such that these values can be used to efficiently and accurately identify similarities between raw signals and a reference genome. To this end, RawHash generates hash values using quantized values of events in two steps, as shown in Figure 5. First, to avoid finding a large number of matches, RawHash uses the quantized values of n consecutive events to pack them in $n \times (Q-p)$ bits while preserving the order information of these consecutive events. RawHash uses several consecutive events in a single hash value because matching a single event is likely to generate a larger number of matches for larger genomes as a single event usually corresponds to a k-mer of 6 to 9 bases depending on the nanopore model [13]. It is essential to use several consecutive events to reduce the number of matching regions between raw signals and the reference genome by increasing the region that these consecutive events span.

Second, to efficiently and accurately find matches between large regions of raw signals and a reference genome using a constrained space, RawHash uses a low collision hash function to generate a 32-bit hash value from $n \times (Q-p)$ bits of n consecutive quantized event values. Since $n \times (Q-p)$ can be larger than 32, using such a hash function is likely to increase the collision rate for dissimilar regions. To avoid inaccurate similarity identifications due to these incorrect collisions, RawHash requires several matches of hash values within close proximity for similarity identification, which we explain next.

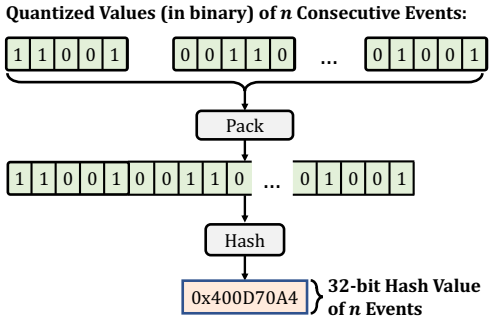

Figure 5: Generating a hash value from n consecutive quantized event values.

2.5. Seeding and Mapping

To efficiently identify similarities, RawHash uses hash values generated from raw nanopore signals and the reference genome in two steps. First, RawHash efficiently identifies matching regions between raw nanopore signals and a reference genome by matching their hash values. These hash values used for matching are usually known as *seeds*. Matching seeds enable efficiently finding similar regions between raw nanopore signals and a reference genome Second, RawHash uses the chaining algorithm proposed in Sigmap [6] to identify the *best* colinear matching seeds within close proximity to identify similar regions between raw signals and a reference genome by chaining. Incorrect seed matches due to collisions or our quantization mechanism that may generate the same quantized value for distinctly dissimilar events are likely to be filtered in the chaining step due to the difficulty of finding colinear seed matches in dissimilar regions. We note that we modify the original chaining algorithm in Sigmap by disabling the distance coefficient as RawHash does not calculate the distance between seed matches, unlike Sigmap.

To efficiently map raw signals to a reference genome, RawHash provides efficient data structures. To this end, RawHash uses hash tables to store the hash values generated from reference genomes (i.e., the indexing step) and efficiently query the same hash table with the hash values generated from the raw signal as the read is sequenced from a nanopore to find positions in the reference genome with matching hash values. RawHash uses the events in chunks to find seed matches and perform chaining in a streaming fashion such that the chaining computation from previous chunks (i.e., seed matches) is transferred to the next chunk if the mapping is unsuccessful for the current chunk.

3. Results

3.1. Evaluation Methodology

We implement RawHash as a tool for mapping raw nanopore signals to a reference genome. Similar to regular read mapping tools, RawHash has two steps to complete the mapping process: 1) indexing the reference genome and 2) mapping raw signals. Although indexing is usually a one-time task that can be performed prior to the mapping step, the indexing of RawHash is efficient that can be performed relatively quickly within a few minutes for large genomes. RawHash provides the mapping information using a standard pairwise mapping format (PAF). In our implementation, we provide an extensive set of parameters that allow configuring several options to fit RawHash for many other applications and nanopore models that we do not evaluate, such as configuring details about the nanopore model (e.g., number of bases per second), number of events that can be included in a single hash value, range of bits to quantize, enabling seeding techniques such as minimizers and fuzzy seed matching. We also provide a default set of parameters that we empirically choose for each common application of real-time genome analysis. These default parameters are set to accurately and efficiently analyze 1) very small (e.g., viral) genomes, 2) small and mid-sized genomes (i.e., genomes with less than a few hundred million bases), 3) large genomes (e.g., genomes with a few billion bases such as a human genome).

We evaluate RawHash in terms of its performance, accuracy, and estimated benefits in sequencing time and cost compared to two state-of-the-art tools UNCALLED and Sigmap. For performance, we evaluate the throughput of each tool in terms of the number of bases they can process per second. Throughput determines if the tool is at least as fast as the speed of DNA passing through a nanopore. For many nanopore models (e.g., R9.4), a DNA strand passes through a pore at around 450 bases per second [3, 6]. It is essential to provide a throughput higher than the throughput of the nanopore to enable real-time genome

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

analysis. To calculate the throughput, we use the tool that UNCALLED provides, `UNCALLED pafstats`, which measures the throughput of the tool from the number of bases that the tool processes and the time it takes to process those bases. Although theoretically, it is not possible to exceed the throughput of a nanopore due to the speed of raw signal generation, for comparison purposes, such a limitation is ignored by `UNCALLED pafstats`.

For accuracy, we evaluate the correctness of the mapping positions that each tool provides when compared to the ground truth mapping positions. To generate the ground truth mapping, we use a read mapping tool, minimap2 [14], to map the basecalled sequences of raw nanopore signals to their corresponding *whole-genome* references. We use `UNCALLED pafstats` to compare the mapping output of a tool with the ground truth mapping to find the number of true positives or *TP* (i.e., correct mappings), false positives or *FP* (i.e., incorrect mappings), and false negatives or *FN* (i.e., unmapped reads that are mapped in ground truth). Correct and incorrect mappings are identified based on the distance of the mapping positions between ground truth and the tool. To evaluate the accuracy, we calculate the precision ($P = TP/(TP+FP)$), recall ($R = TP/(TP+FN)$) and the F_1 ($F_1 = 2 \times (P \times R)/(P+R)$) values.

For estimating the benefits in sequencing time and cost of each tool, we calculate the average time that each tool takes to process each read for mapping. The mean time determines the estimations of how quickly each tool can make a mapping decision to activate Read Until, which indicates the potential savings from overall sequencing time and cost.

We evaluate RawHash, UNCALLED, and Sigmap for three applications 1) read mapping, 2) relative abundance estimation, and 3) contamination analysis. Read mapping aims to map the raw signals to their corresponding reference genomes. Relative abundance estimation measures the abundance of each genome relative to other genomes in the same sample by mapping raw signals to a given set of reference genomes. Contamination analysis aims to identify if a sample is contaminated with a certain genome (e.g., a viral genome) by mapping raw signals to the reference genome that the sample may be contaminated with. For each tool, we use their default parameter settings in our evaluation.

To evaluate each of these applications, we use real datasets that we list in Table 1. These datasets include both raw nanopore signals in the FAST5 [21] format and their corresponding basecalled sequences in the FASTA format [22]. For relative abundance estimation, we create a mock community using all the read sets from datasets D1 to D5, and the reference genome is the combination of reference genomes used in these datasets. We slightly modify the reference genome we use in the relative abundance estimation such that the sequence IDs in the reference genome provide additional information about the species (e.g., taxonomy IDs) to enable calculating relative abundance in real-time. For contamination analysis, we combine the SARS-CoV-2 read sets (D1) with human read sets (D5) to identify if the combined sample is contaminated with the SARS-CoV-2 sample by mapping raw signals in the combined set to the SARS-CoV-2 reference genome. For all evaluations, we use the AMD EPYC 7742 processor at 2.26GHz to run the tools.

Evaluating Sequence Until. Our goal is to avoid redundant sequencing to reduce sequencing time and cost for relative abundance estimation. We find that the Read Until mechanism can be utilized to *fully* stop the sequencing run when the real-time relative abundance estimation reaches a certain confidence level to achieve accurate estimations, which we call *Sequence Until*. While a similar mechanism is evaluated to enrich the coverage depth of low-abundance species [5], we evaluate the potential benefits of Read Until for low-cost relative abundance estimations. We integrate a real-time confidence calculation mechanism in RawHash to activate the Sequence Until mechanism in three steps. First, RawHash measures the relative abundance estimation after every n reads that can be mapped to a reference genome in real-time. Second, to identify if the recently mapped reads provide substantial changes in the abundance estimations, RawHash performs a cross-correlation calculation between the last w estimations. Cross-correlation can identify *outliers* from a set of estimations to identify if the outlier is substantially different than other estimations, which indicates that recent reads can still change the relative abundance estimation, and more reads should be sequenced from the sample. Third, RawHash activates Sequence Until by fully stopping the sequencing (e.g., continuous use of Read Until to instantly eject reads without processing them) when there are no outliers in the last w estimations, which indicates a convergence to a certain relative abundance estimation, and further sequencing is unlikely to change this estimation. RawHash provides a set of parameters to adjust these parameters related to Sequence Until.

Table 1: Details of datasets used in our evaluation.

	Organism	Reads (#)	Bases (#)	SRA Accession	Reference Genome	Genome Size
			Read Mapping			
D1	*SARS-CoV-2*	1,382,016	594M	CADDE [23]	GCF_009858895.2	29,903
D2	*E. coli*	353,317	2,365M	ERR9127551	GCA_000007445.1	5M
D3	*Yeast*	49,989	380M	SRR8648503	GCA_000146045.2	12M
D4	*Green Algae*	29,933	609M	ERR3237140	GCF_000002595.2	111M
D5	*Human HG001*	269,507	1,584M	FAB42260 [24]	T2T-CHM13 (v2)	3,117M
			Relative Abundance Estimation			
	D1-D5	2,084,762	5,531M	D1-D5	D1-D5	3,246M
			Contamination Analysis			
	D1, D5	1,651,523	2,178M	D1, D5	D1	29,903

Dataset numbers (e.g., D1-D5) show the combined datasets. Base counts in millions (M).

We evaluate the benefits of Sequence Until by comparing 1) RawHash without Sequence Until and 2) RawHash with Sequence Until in terms of 1) the difference in the relative abundance estimations and 2) the estimated benefits in sequencing time and cost. To evaluate Sequence Until in a realistic sequencing environment where reads from different species can be sequenced in a random order, we randomly shuffle the reads in the relative abundance dataset and generate a set of 50,000 reads with a random order of species so that we can simulate this random behavior. We also find that Sequence Until can be applied to other mechanisms. To evaluate the potential benefits of Sequence Until, we simulate the benefits when using UNCALLED with Sequence Until and compare it with RawHash.

3.2. Performance

Figure 6 shows the throughput of regular nanopores that we use as a baseline and the throughput of the tools when mapping raw nanopore signals to each dataset for read mapping, contamination analysis, and relative abundance estimation. We make three key observations. First, RawHash and UNCALLED are the *only* tools that can perform real-time genome analysis for large genomes, as they can provide higher throughputs than nanopores for all datasets. Sigmap *cannot* perform real-time

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

genome analysis for large genomes as it can provide 0.7× and 0.6× throughput of a nanopore for human genome mapping and relative abundance estimations, respectively. RawHash can achieve high throughput as its seeding mechanism is based on efficiently matching hash values compared to the costly distance calculations that Sigmap performs for matching seeds, which shows poor scalability for larger genomes. Second, the throughput of UNCALLED is not affected by the genome size as it provides a near-constant throughput of around 16× for all applications. This is because UNCALLED uses FM-index [25] and a branching algorithm that provides robust scaling with respect to the reference genome size [3]. Third, RawHash provides an average throughput 25.8× and 3.4× better than UNCALLED and Sigmap, respectively. We conclude that RawHash provides significant benefits in improving the throughput to scale the real-time analysis to large genomes while matching the throughput of nanopores.

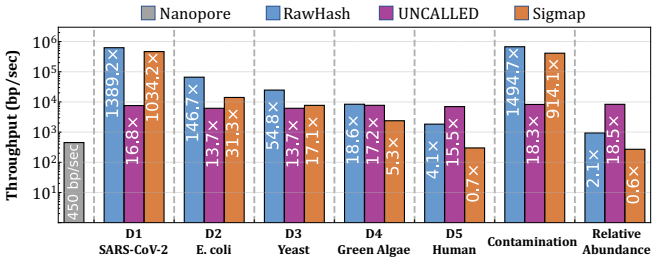

Figure 6: Throughput of each tool. Values inside the bars show the throughput ratio between each tool and a nanopore.

3.3. Accuracy

Table 2 shows the accuracy results of tools for each dataset and application. We make four key observations. First, RawHash provides the best accuracy in terms of precision, recall, and F_1 values compared to UNCALLED and Sigmap when mapping reads to large genomes (i.e., the human genome and the relative abundance estimation). RawHash can efficiently match several events using hash values, which is specifically beneficial in reducing the number of matching regions in large genomes and increasing the specificity due to finding longer matches compared to UNCALLED and Sigmap.

Second, RawHash and UNCALLED can accurately perform contamination analysis while Sigmap suffers from significantly lower precision and recall values. Due to the nature of a contamination analysis, it is essential to correctly eliminate the genomes other than the contaminating genome (precision) without missing the correct mappings of reads from the contaminating genome (recall). Unfortunately, Sigmap can provide high values in neither of these categories, making it significantly unsafe for contamination detection.

Third, the precision of RawHash does not drop with the increased length in the reference genome due to the benefits of finding long matches, which provides a higher confidence in read mapping.

Fourth, although RawHash does not provide the best accuracy when mapping reads to genomes smaller than the human genome, its accuracy is on par with UNCALLED and Sigmap for these genomes. UNCALLED and Sigmap can achieve high recall values as their mechanisms are best optimized for accurately handling matches in relatively smaller genomes with fewer repeats and ambiguous mappings [3,6]. We conclude that RawHash is the *only* tool that can accurately scale to performing real-time genome analysis for large genomes, especially with significantly high precision rates.

Table 2: Mapping accuracy.

Dataset		UNCALLED	Sigmap	RawHash
	Read Mapping			
D1	Precision	0.9547	**0.9929**	0.9868
SARS-CoV-2	Recall	**0.9910**	0.5540	0.8735
	F_1	**0.9725**	0.7112	0.9267
D2	Precision	0.9816	**0.9842**	0.9573
E. coli	Recall	**0.9647**	0.9504	0.9009
	F_1	**0.9731**	0.9670	0.9282
D3	Precision	0.9459	0.9856	**0.9862**
Yeast	Recall	**0.9366**	0.9123	0.8412
	F_1	0.9412	**0.9475**	0.9079
D4	Precision	0.8836	**0.9741**	0.9691
Green Algae	Recall	0.7778	**0.8987**	0.7015
	F_1	0.8273	**0.9349**	0.8139
D5	Precision	0.4867	0.4287	**0.8959**
Human HG001	Recall	0.2379	0.2641	**0.4054**
	F_1	0.3196	0.3268	**0.5582**
	Relative Abundance Estimation			
	Precision	0.7683	0.7928	**0.9484**
D1-D5	Recall	0.1273	0.2739	**0.3076**
	F_1	0.2184	0.4072	**0.4645**
	Contamination Analysis			
	Precision	**0.9378**	0.7856	0.8733
D1, D5	Recall	**0.9910**	0.5540	0.8735
	F_1	**0.9637**	0.6498	0.8734

Best results are highlighted with **bold** text.

Relative Abundance Estimations. Table 3 shows the relative abundance estimations that each tool makes and the Euclidean distance of their estimation to the ground truth estimation. We make two key observations. First, we find that RawHash provides the most accurate relative abundance estimations in terms of the estimation distance to the ground truth compared to UNCALLED and Sigmap. This observation correlates with the accuracy results we show in Table 2 where RawHash provides the best overall accuracy for relative estimation, which results in generating the most accurate relative abundance estimations. Second, although Sigmap cannot perform real-time relative abundance estimation due to its throughput being lower than a nanopore (Figure 6), Sigmap provides accurate estimations that are on par with RawHash. This observation shows that while Sigmap provides mappings with more incorrect positions due to lower precision than RawHash (Table 2), these reads with incorrect mapping positions are mostly mapped to their correct species. We conclude that RawHash is the *only* tool that can *accurately* be applied to analyze relative abundance estimations while matching the throughput of nanopores at a large-scale based on the prior knowledge of the set of reference genomes to map the reads.

3.4. Sequencing Time and Cost

Our goal is to estimate the benefits that each tool provides in reducing the sequencing time and cost. To this end, we measure the average time each tool takes for each read while the read is sequenced in a nanopore, as shown in Figure 7. We make two

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

Table 3: Relative abundance estimations.

	Estimated Relative Abundance Ratios					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
Ground Truth	0.0929	0.4365	0.0698	0.1179	0.2828	N/A
UNCALLED	0.0026	0.5884	0.0615	0.1313	0.2161	0.1895
Sigmap	0.0419	0.4191	0.1038	0.0962	0.3390	0.0877
RawHash	0.1249	0.4701	0.0957	0.0629	0.2464	**0.0847**

Best results are highlighted with **bold** text.

key observations. First, RawHash provides an average speedup of 32.1× and 2.1× per read compared to UNCALLED and Sigmap, respectively. This observation suggests that RawHash can 1) reduce the number of overall bases that need to be sequenced in the entire sequencing lifetime (sequencing cost) and 2) increase the number of bases sequenced at a constant time interval by utilizing Read Until more frequently than other tools. Second, while RawHash and Sigmap are significantly faster than UNCALLED for shorter reference genomes, UNCALLED usually spends less time per read when mapping them to larger genomes. This observation correlates with our observation in the near-constant throughput results of UNCALLED, which shows that the performance of UNCALLED is not significantly affected by the reference genome length. We conclude that RawHash can provide significant reductions in sequencing time and cost as it, on average, takes shorter times when mapping reads, while UNCALLED is better at reducing the time and cost for larger genomes.

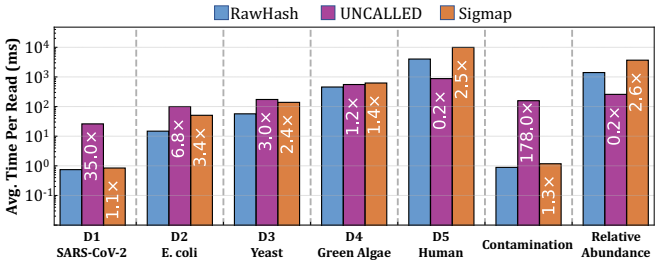

Figure 7: Average time spent per read by each tool in real-time. Values inside the bars show the speedups that RawHash provides over other tools in each dataset.

3.5. Benefits of Sequence Until

Simulated Sequence Until. Our goal is to estimate the benefits of implementing the Sequence Until mechanism in UNCALLED and compare it with RawHash when they both use Sequence Until under the same conditions. To this end, we use `shuf` in Linux to randomly shuffle the mapping files that both RawHash and UNCALLED generate for relative abundance and extract a certain portion of the randomly shuffled file to identify their relative abundance estimations after 0.01%, 0.1%, 1%, 10%, and 25% of the overall reads in the sample are randomly sequenced from nanopores.

Table 4 shows the distance of relative abundance estimations after a certain portion of the read is randomly sequenced from nanopores. We make two key observations. First, *both* RawHash and UNCALLED can significantly benefit from Sequence Until by stopping sequencing after processing a smaller portion of the entire sample since their estimations using smaller portions are close to those using the entire set of reads (Table 3) in terms of their distance to the ground truth. This suggests that many other tools can benefit from Sequence Until as their sensitivity to relative abundance estimations may not significantly change while providing opportunities for reducing the sequencing time and cost up to a certain threshold based on the tool.

Second, RawHash can provide more accurate relative abundance estimations when using only 0.1% of the reads than the estimation that UNCALLED provides using the entire set of reads (Table 3). We conclude that Sequence Until provides significant opportunities in reducing sequencing time and cost while more accurate tools such as RawHash can benefit further from Sequence Until by using fewer portions of the entire read set than the portions that less accurate tools would need to achieve similar accuracy.

Table 4: Relative abundance with simulated Sequence Until.

	Estimated Relative Abundance Ratios					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
Ground Truth	0.0929	0.4365	0.0698	0.1179	0.2828	N/A
UNCALLED (25%)	0.0026	0.5890	0.0613	0.1332	0.2139	0.1910
RawHash (25%)	0.0271	0.4853	0.0920	0.0786	0.3170	**0.0995**
UNCALLED (10%)	0.0026	0.5906	0.0611	0.1316	0.2141	0.1920
RawHash (10%)	0.0273	0.4869	0.0963	0.0772	0.3124	**0.1004**
UNCALLED (1%)	0.0026	0.5750	0.0616	0.1506	0.2103	0.1836
RawHash (1%)	0.0259	0.4783	0.0987	0.0882	0.3088	**0.0928**
UNCALLED (0.1%)	0.0040	0.4565	0.0380	0.1910	0.3105	0.1242
RawHash (0.1%)	0.0212	0.5045	0.1120	0.0810	0.2814	**0.1136**
UNCALLED (0.01%)	0.0000	0.5551	0.0000	0.0000	0.4449	0.2602
RawHash (0.01%)	0.0906	0.6122	0.0000	0.0000	0.2972	**0.2232**

Percentages show the portion of the overall reads used. Best results are highlighted with **bold** text.

Sequence Until with RawHash. Our goal is to evaluate Sequence Until when used in real-time with RawHash for relative abundance estimation. Table 5 shows the relative abundance estimations that RawHash makes with and without Sequence Until. We note that the estimations we show for RawHash in Table 5 are different than the estimations in Table 3 since we randomly subsample the reads in the relative abundance estimation dataset, as explained in Section 3.1. We make two key observations. First, we observe that the distance between the relative abundance estimations between these two configurations of RawHash is substantially low. This indicates that our outlier detection mechanism can accurately detect the convergence to the relative abundance estimations without using a full set of reads. Second, Sequence Until enables accurately stopping the entire sequencing after processing 7% of the reads in the entire set without substantially sacrificing accuracy. We conclude that Sequence Until has the potential to significantly reduce the sequencing time and cost by using only fewer reads from a sample while producing accurate results.

Table 5: Relative abundance with Sequence Until.

	Estimated Relative Abundance Ratios in 50,000 Random Reads					
Tool	***SARS-CoV-2***	***E. coli***	***Yeast***	***Green Algae***	***Human***	**Distance**
RawHash (100%)	0.0270	0.3636	0.3062	0.1951	0.1081	N/A
RawHash + Sequence Until (7%)	0.0283	0.3539	0.3100	0.1946	0.1133	0.0118

Percentages show the portion of the overall reads used.

4. Discussion

We discuss the benefits we expect RawHash can immediately make, the limitations of RawHash, and future work. We envision that RawHash can be useful mainly for two directions. First, RawHash provides a low-cost solution for analyzing large genomes in real-time. Such an analysis can be significantly useful when using nanopore sequencers with limited computa-

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

tional resources to enable portable real-time genome analysis at a large scale.

Second, we expect that RawHash can also be useful for genome analysis that does not require real-time solutions by reducing the time and energy that further steps in genome analysis may require. One of the immediate steps after generating raw nanopore signals is their translation to their corresponding DNA bases as sequences of characters with a computationally-intensive step, *basecalling*. Basecalling approaches are usually computationally costly and consume significant energy as they use complex deep learning models [26–38]. Although we do not evaluate in this work, we expect that RawHash can be used as a low-cost filter to eliminate the reads that are unlikely to be useful in downstream analysis, which can reduce the overall workload of basecallers and further downstream analysis. We believe that RawHash can be applied for such a filtering purpose since a previous work [39] proposes a lightweight basecalling solution to use as a filter before using the costly basecallers.

Limitations. We find four limitations of RawHash, which we believe can be improved with further optimizations and better solutions. First, RawHash depends on previously generated k-mer models to generate events from reference genomes. Although these k-mer models can be trained and generated [17,18], this makes it challenging to adapt the most accurate parameters for each k-mer model based on the nanopore model used for sequencing. A more generic k-mer model that can accurately represent all nanopores is needed to easily adapt RawHash to all possible nanopore models that may be released in the future.

Second, RawHash starts providing lower recall values as the genome size increases, which indicates that a larger portion of correct reads cannot be mapped by RawHash due to the increase in the number of false negatives. Although such an increase in false negatives does not substantially affect some applications, such as contamination analysis, where providing higher precision is more critical to correctly identify the contaminated sample, improving it is useful to provide more accurate genome analysis overall.

Third, we perform our relative abundance estimations based on a priori knowledge of reference genomes. While such an experiment can still be useful in practical scenarios, this is not the common case in metagenomic analysis, where a sample is searched against a significantly larger set of species. We expect that our mechanism can still scale to such metagenomic analyses given that many metagenomic databases are efficiently constructed by including fewer and useful information for each species [40], as opposed to our analysis, where we include whole-genome references.

Fourth, we observe that the throughput of RawHash is expected to reach the throughput of a nanopore when analyzing reference genomes slightly larger than a human genome. Such a limitation can be alleviated by applying 1) seeding techniques that provide faster and more space-efficient searches in large spaces and 2) chaining algorithms that are optimized for hash-based seed matches without the notion of distance between seeds, unlike the chaining algorithm used in Sigmap.

Future work. We find three key directions for future work. First, we find that our efficient hash-based similarity identification mechanism can be used to efficiently find overlaps between signals as the reads are sequenced in real-time. Although we observe that our indexing technique is efficient in terms of the amount it requires to construct an index even for large genomes, such an overlapping technique requires substantially more optimized indexing methods and techniques that can efficiently find overlaps as more reads are sequenced and *evolves* the index. Finding overlaps between signals can be beneficial in 1) providing enriched information to basecallers to increase their basecalling accuracy and 2) identifying *redundant* signals that fully overlap with already sequenced reads in a potential effort for generating assemblies from signals.

Second, since RawHash generates hash values for matching similar regions, it provides opportunities to use the hash-based seeding techniques [14, 41–70] that are optimized for identifying sequence similarities accurately without requiring large memory space, such as minimizers [14, 71], spaced seeds [45], syncmers [67], strobemers [68, 69], and fuzzy seed matching as in BLEND [70]. Although we do not evaluate in this work, we implement the minimizer seeding technique in RawHash. Our initial observation motivates us that future work can exploit these seeding techniques with slight modifications in their seeding mechanisms to significantly improve the performance of certain applications without reducing the accuracy.

Third, we find that RawHash can also benefit from a GPU implementation as its low-cost and accurate implementation can effectively be scaled to nanopore sequencers that include thousands of nanopores such that these pores can be analyzed in parallel with an efficient GPU implementation, which we leave as future work.

5. Related Work

To our knowledge, *RawHash* is the first mechanism to efficiently and accurately perform real-time analysis of raw nanopore signals for large genomes. We discuss related work in 1) basecalling, 2) accelerating genome analysis after the basecalling step, and 3) real-time genome analysis with limited computational resources.

Basecalling. Deep learning-based models are utilized by modern basecallers to considerably enhance the precision of identifying a nucleotide base from raw signals compared to traditional non-deep learning-based basecallers [26, 27, 72–75]. Deep learning models can successfully basecall genomes due to the developments and advancements in their architecture, which enables them to model and accurately recognize spatial characteristics in the raw data. Many basecallers have been proposed using modern deep learning-based architectures [28–37] However, the use of complex deep learning models makes basecalling slow and memory-hungry, bottlenecking all genomic analyses that depend on it [27]. Recent works focus on developing methods to speed up the basecalling process. One approach to basecalling acceleration is to use specialized hardware, such as field-programmable gate arrays (FPGAs) [76–80] or processing-in-memory (PIM) [38, 81, 82], to perform the basecalling computations. These specialized hardware devices can perform many calculations in parallel, allowing for significant speedups in the basecalling process. Another approach is to use machine learning-based compression techniques to improve the performance of the basecalling process. RUBICON [27] provides a framework to develop hardware-optimized basecallers using neural architecture search [83], knowledge distillation [84], and pruning [85]. Dorado [86], a basecaller by ONT, uses quantization [87] to reduce the bit-width precision at which neural network calculations are performed. All the above works accelerate the basecalling step without eliminating the

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

wasted computation in basecalling. TargetCall [39] proposes a pre-basecalling filter that eliminates the wasted computation in basecalling by leveraging the observation that the majority of reads are discarded after basecalling. However, RawHash is different from these works as its goal is to perform real-time analysis of raw signals without performing the computationally-intensive basecalling step.

Accelerating the genome analysis after basecalling. There are several works that aim to accelerate the entire genome analysis pipeline by accelerating one or multiple steps in the pipeline after basecalling the raw nanopore signals [88, 89]. These works accelerate the pre-alignment filtering and read classification [90–102], chaining [103, 104], read mapping and sequence alignment [105–162] steps. Although these works can significantly improve the performance of the genome analysis pipeline, unlike RawHash, these works cannot perform real-time genome analysis while the raw nanopore signals are generated from nanopore sequencers.

Real-time analysis of raw nanopore signal. Several works perform real-time genome analysis of raw nanopore signals by utilizing Read Until [3, 5–11]. SquiggleFilter [7] uses an ASIC accelerator that quickly filters non-related raw electrical signals before basecalling for viral detection. HARU [10] is an FPGA accelerator that accelerates real-time selective genome sequencing on resource-constrained devices for detecting viral genomes. RawHash differs from these works as it does not require specialized hardware design and can scale to analyze large genomes while matching the throughput of nanopores.

SquiggleNet [9] and RawMap [11] require training with machine learning techniques using sequencing reads as training data without using reference genomes. These works train their models to classify raw nanopore signals without mapping them to the reference genome, which is different than RawHash as it maps raw signals to a reference genome. These works often require retraining and reconfiguring the neural network model and architectures. Although such classification approaches can provide high accuracy in labeling reads as target or non-target reads based on a target genome of interest, it can be challenging to easily perform real-time analysis with high accuracy without retraining or reconfiguring these models. RawHash is different than these works as it can map reads to any reference genome using easily configurable parameter settings.

ReadFish [5] can scale to mapping reads to large genomes such as a human genome using GPUs for performing basecalling. RawHash differs from ReadFish as it does not require powerful computational resources such as GPUs, which may not be immediately available for portable sequences such as ONT MinION. Similar to ReadFish, RUBRIC [8] uses a basecalling approach followed by mapping the basecalled raw signals to analyze raw nanopore signals in real-time. These basecalling approaches are optimized to use the *entire* raw nanopore signal of a read rather than the portions of raw signals produced in real-time, which can be challenging in generating an accurate mapping with a small number of basecalled signals [3, 6]. RawHash differs from ReadFish and RUBRIC as it can directly and accurately map a small number of raw signals (e.g., signals produced in one second) to a reference genome without basecalling them.

UNCALLED [3] and Sigmap [6] are the most relevant works to RawHash. These works map raw nanopore signals to a reference genome without using powerful computational resources (e.g., GPUs), which can be directly used with portable nanopore sequencers. UNCALLED detects events from raw signals, and the probability of k-mers that each event can represent is calculated using k-mer models. UNCALLED identifies the sequence of matching k-mers between the most probable k-mers of events and a reference genome using an FM-index [25]. However, it becomes challenging to accurately identify the matching regions with such a probabilistic model from a large number of matches as the genome size increases. Thus, UNCALLED is highly accurate for small genomes (e.g., *E. coli* and *Yeast* genomes) due to the smaller number of probabilistic matches in the reference genome that can be identified accurately.

Sigmap can map raw nanopore signals to genomes larger than the *Yeast* genome (e.g., *Green Algae* with around 100M bases). To achieve this, Sigmap converts the k-mers of the reference genome into events and matches the events between raw nanopore signals with the events of the reference genome. Since events are not necessarily identical when reading the same DNA content, it is challenging to find accurate matches between them due to the signal variations we discuss in Section 2.2. To address this challenge, Sigmap creates a vector from each n consecutive events (i.e., n-dimensional vector space) from the reference genome (i.e., the indexing step) and measures the Euclidean distance between these vectors and the vectors generated from raw nanopore signals (i.e., the mapping step) using a k-d tree structure. Although the distance between vector of events generated from similar regions is close, such a distance calculation is computationally *costly* and suffers from the *curse of dimensionality* that fundamentally prevents accurately and efficiently increasing the number of events within a single vector, which makes it ineffective for larger genomes. RawHash is different than UNCALLED and Sigmap as it identifies similarities between a reference genome and a raw nanopore signal by efficiently and accurately matching the hash values generated from them without using probabilistic approaches that can be inaccurate for large genomes or costly distance calculations.

6. Conclusion

We propose RawHash, a novel mechanism that provides a low-cost and accurate approach for real-time genome analysis for large genomes. RawHash can efficiently and accurately perform real-time analysis of raw nanopore signals to identify similarities between the signals and a reference genome in real-time at a large-scale (e.g., whole-genome analysis for human or communities with multiple samples). To efficiently and accurately identify similarities, RawHash 1) generates events from both raw signals and the reference genome, 2) quantizes the events into values such that slightly different events that correspond to the same DNA content can have the same value, and 3) generates hash values from multiple events to efficiently find matching regions between raw signals and a reference genome using hash values with efficient data structures such as hash tables. We compare RawHash with the state-of-the-art approaches, UNCALLED and Sigmap, on three important applications in terms of their performance, accuracy, and estimated benefits in reducing sequencing time and cost. Our results show that 1) RawHash is the *only* tool that can be accurately applied to analyze raw nanopore signals at large-scale, 2) provides 25.8× and 3.4× better average throughput, and 3) can use Read Until 32.1× and 2.1× faster than UNCALLED and Sigmap, respectively.

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

References

[1] J. Shendure, S. Balasubramanian, G. M. Church, W. Gilbert, J. Rogers, J. A. Schloss, and R. H. Waterston, "DNA sequencing at 40: past, present and future," *Nature*, vol. 550, Oct. 2017.

[2] M. Jain, S. Koren, K. H. Miga, J. Quick, A. C. Rand, T. A. Sasani, J. R. Tyson, A. D. Beggs, A. T. Dilthey, I. T. Fiddes, S. Malla, H. Marriott, T. Nieto, J. O'Grady, H. E. Olsen, B. S. Pedersen, A. Rhie, H. Richardson, A. R. Quinlan, T. P. Snutch, L. Tee, B. Paten, A. M. Phillippy, J. T. Simpson, N. J. Loman, and M. Loose, "Nanopore sequencing and assembly of a human genome with ultra-long reads," *Nature Biotechnology*, vol. 36, Apr. 2018.

[3] 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, Apr. 2021.

[4] M. Loose, S. Malla, and M. Stout, "Real-time selective sequencing using nanopore technology," *Nature Methods*, vol. 13, Sep. 2016.

[5] A. Payne, N. Holmes, T. Clarke, R. Munro, B. J. Debebe, and M. Loose, "Readfish enables targeted nanopore sequencing of gigabase-sized genomes," *Nature Biotechnology*, vol. 39, Apr. 2021.

[6] H. Zhang, H. Li, C. Jain, H. Cheng, K. F. Au, H. Li, and S. Aluru, "Real-time mapping of nanopore raw signals," *Bioinformatics*, vol. 37, Jul. 2021.

[7] T. Dunn, H. Sadasivan, J. Wadden, K. Goliya, K.-Y. Chen, D. Blaauw, R. Das, and S. Narayanasamy, "SquiggleFilter: An accelerator for portable virus detection," in *MICRO*, 2021.

[8] H. S. Edwards, R. Krishnakumar, A. Sinha, S. W. Bird, K. D. Patel, and M. S. Bartsch, "Real-Time Selective Sequencing with RUBRIC: Read Until with Basecall and Reference-Informed Criteria," *Scientific Reports*, vol. 9, Aug. 2019.

[9] Y. Bao, J. Wadden, J. R. Erb-Downward, P. Ranjan, W. Zhou, T. L. McDonald, R. E. Mills, A. P. Boyle, R. P. Dickson, D. Blaauw, and J. D. Welch, "SquiggleNet: real-time, direct classification of nanopore signals," *Genome Biology*, vol. 22, Oct. 2021.

[10] P. J. Shih, H. Saadat, S. Parameswaran, and H. Gamaarachchi, "Efficient real-time selective genome sequencing on resource-constrained devices," *arXiv*, Nov. 2022.

[11] 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," *bioRxiv*, Jan. 2023.

[12] B. Noordijk, R. Nijland, V. J. Carrion, J. M. Raaijmakers, D. de Ridder, and C. de Lannoy, "baseLess: lightweight detection of sequences in raw MinION data," *bioRxiv*, Jan. 2022.

[13] M. David, L. J. Dursi, D. Yao, P. C. Boutros, and J. T. Simpson, "Nanocall: an open source basecaller for Oxford Nanopore sequencing data," *Bioinformatics*, vol. 33, Jan. 2017.

[14] H. Li, "Minimap2: pairwise alignment for nucleotide sequences," *Bioinformatics*, vol. 34, Sep. 2018.

[15] M. I. Abouelhoda and E. Ohlebusch, "Chaining algorithms for multiple genome comparison," *Combinatorial Pattern Matching (CPM) Special Issue*, vol. 3, Jun. 2005.

[16] Oxford Nanopore Technologies, "K-mer models, https://github.com/nanoporetech/kmer_models/."

[17] J. T. Simpson, R. E. Workman, P. C. Zuzarte, M. David, L. J. Dursi, and W. Timp, "Detecting DNA cytosine methylation using nanopore sequencing," *Nature Methods*, vol. 14, Apr. 2017.

[18] Oxford Nanopore Technologies, "Nanopolish (K-mer models), https://github.com/jts/nanopolish/tree/r10/etc/r10-models."

[19] Oxford Nanopore Technologies, "Scrappie, https://github.com/nanoporetech/scrappie."

[20] G. D. Ruxton, "The unequal variance t-test is an underused alternative to Student's t-test and the Mann–Whitney U test," *Behavioral Ecology*, vol. 17, Jul. 2006.

[21] A. Payne, N. Holmes, V. Rakyan, and M. Loose, "BulkVis: a graphical viewer for Oxford nanopore bulk FAST5 files," *Bioinformatics*, vol. 35, Jul. 2019.

[22] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison." *Proceedings of the National Academy of Sciences*, vol. 85, Apr. 1988.

[23] CADDE Centre, "SARS-CoV-2 Dataset, https://cadde.s3.climb.ac.uk/SP1-raw.tgz."

[24] Nanopore WGS Consortium, "Whole Human Genome Sequencing Dataset (NA12878), https://github.com/nanopore-wgs-consortium/NA12878/blob/master/Genome.md."

[25] P. Ferragina and G. Manzini, "Opportunistic data structures with applications," in *Proceedings 41st Annual Symposium on Foundations of Computer Science*, 2000.

[26] D. Senol Cali, J. S. Kim, S. Ghose, C. Alkan, and O. Mutlu, "Nanopore Sequencing Technology and Tools for Genome Assembly: Computational Analysis of the Current State, Bottlenecks and Future Directions," *Briefings in Bioinformatics*, vol. 20, Jul. 2019.

[27] G. Singh, M. Alser, A. Khodamoradi, K. Denolf, C. Firtina, M. B. Cavlak, H. Corporaal, and O. Mutlu, "A framework for designing efficient deep learning-based genomic basecallers," *bioRxiv*, 2022.

[28] Oxford Nanopore Technologies, "Bonito, https://github.com/nanoporetech/bonito."

[29] H. Konishi, R. Yamaguchi, K. Yamaguchi, Y. Furukawa, and S. Imoto, "Halcyon: an accurate basecaller exploiting an encoder-decoder model with monotonic attention," *Bioinformatics*, 2021.

[30] N. Huang, F. Nie, P. Ni, F. Luo, and J. Wang, "SACall: A Neural Network Basecaller for Oxford Nanopore Sequencing Data Based on Self-Attention Mechanism," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2020.

[31] Z. Xu, Y. Mai, D. Liu, W. He, X. Lin, C. Xu, L. Zhang, X. Meng, J. Mafofo, W. A. Zaher *et al.*, "Fast-bonito: A Faster Deep Learning Based Basecaller for Nanopore Sequencing," *Artificial Intelligence in the Life Sciences*, vol. 1, 2021.

[32] V. Boža, B. Brejová, and T. Vinař, "DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads," *PLOS One*, 2017.

[33] Oxford Nanopore Technologies, "Guppy."

[34] P. Perešíni, V. Boža, B. Brejová, and T. Vinař, "Nanopore base calling on the edge," *Bioinformatics*, 2021.

[35] X. Lv, Z. Chen, Y. Lu, and Y. Yang, "An end-to-end Oxford nanopore basecaller using convolution-augmented transformer," in *BIBM*, 2020.

[36] J. Zeng, H. Cai, H. Peng, H. Wang, Y. Zhang, and T. Akutsu, "Causalcall: Nanopore basecalling using a temporal convolutional network," *Frontiers in Genetics*, 2020.

[37] Y.-M. Yeh and Y.-C. Lu, "MSRCall: A multi-scale deep neural network to basecall Oxford nanopore sequences," *Bioinformatics*, 2022.

[38] H. Mao, M. Alser, M. Sadrosadati, C. Firtina, A. Baranwal, D. S. Cali, A. Manglik, N. A. Alserr, and O. Mutlu, "Genpip: In-memory acceleration of genome analysis via tight integration of basecalling and read mapping," in *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 2022.

[39] M. B. Cavlak, G. Singh, M. Alser, C. Firtina, J. Lindegger, M. Sadrosadati, N. M. Ghiasi, C. Alkan, and O. Mutlu, "Targetcall: Eliminating the wasted computation in basecalling via pre-basecalling filtering," *bioRxiv*, 2022.

[40] F. P. Breitwieser, D. N. Baker, and S. L. Salzberg, "KrakenUniq: confident and fast metagenomics classification using unique k-mer counts," *Genome Biology*, vol. 19, Nov. 2018.

[41] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, "Basic local alignment search tool," *Journal of Molecular Biology*, vol. 215, Oct. 1990.

[42] S. F. Altschul, T. L. Madden, A. A. Schäffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Research*, vol. 25, Sep. 1997.

[43] Z. Ning, A. J. Cox, and J. C. Mullikin, "SSAHA: A Fast Search Method for Large DNA Databases," *Genome Research*, vol. 11, Oct. 2001.

[44] W. J. Kent, "BLAT—The BLAST-Like Alignment Tool," *Genome Research*, vol. 12, Apr. 2002.

[45] B. Ma, J. Tromp, and M. Li, "PatternHunter: faster and more sensitive homology search," *Bioinformatics*, vol. 18, Mar. 2002.

[46] S. Schwartz, W. J. Kent, A. Smit, Z. Zhang, R. Baertsch, R. C. Hardison, D. Haussler, and W. Miller, "Human–Mouse Alignments with BLASTZ," *Genome Research*, vol. 13, Jan. 2003.

[47] G. S. C. Slater and E. Birney, "Automated generation of heuristics for biological sequence comparison," *BMC Bioinformatics*, vol. 6, Feb. 2005.

[48] T. D. Wu and C. K. Watanabe, "GMAP: a genomic mapping and alignment program for mRNA and EST sequences," *Bioinformatics*, vol. 21, May 2005.

[49] B. D. Ondov, A. Varadarajan, K. D. Passalacqua, and N. H. Bergman, "Efficient mapping of Applied Biosystems SOLiD sequence data to a reference genome for functional genomic applications," *Bioinformatics*, vol. 24, Dec. 2008.

[50] R. Li, Y. Li, K. Kristiansen, and J. Wang, "SOAP: short oligonucleotide alignment program," *Bioinformatics*, vol. 24, Mar. 2008.

[51] H. Jiang and W. H. Wong, "SeqMap: mapping massive amount of oligonucleotides to the genome," *Bioinformatics*, vol. 24, Oct. 2008.

[52] H. Lin, Z. Zhang, M. Q. Zhang, B. Ma, and M. Li, "ZOOM! Zillions of oligos mapped," *Bioinformatics*, vol. 24, Nov. 2008.

[53] A. D. Smith, Z. Xuan, and M. Q. Zhang, "Using quality scores and longer reads improves accuracy of Solexa read mapping," *BMC Bioinformatics*, vol. 9, Feb. 2008.

[54] C. Alkan, J. M. Kidd, T. Marques-Bonet, G. Aksay, F. Antonacci, F. Hormozdiari, J. O. Kitzman, C. Baker, M. Malig, O. Mutlu, S. C. Sahinalp, R. A. Gibbs, and E. E. Eichler, "Personalized copy number and segmental duplication maps using next-generation sequencing," *Nature Genetics*, vol. 41, Oct. 2009.

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[55] N. Homer, B. Merriman, and S. F. Nelson, "BFAST: An Alignment Tool for Large Scale Genome Resequencing," *PLOS ONE*, vol. 4, Nov. 2009.

[56] K. Schneeberger, J. Hagmann, S. Ossowski, N. Warthmann, S. Gesing, O. Kohlbacher, and D. Weigel, "Simultaneous alignment of short reads against multiple genomes," *Genome Biology*, vol. 10, Sep. 2009.

[57] D. Weese, A.-K. Emde, T. Rausch, A. Döring, and K. Reinert, "RazerS—fast read mapping with sensitivity control," *Genome Research*, vol. 19, Sep. 2009.

[58] S. M. Rumble, P. Lacroute, A. V. Dalca, M. Fiume, A. Sidow, and M. Brudno, "SHRiMP: Accurate Mapping of Short Color-space Reads," *PLOS Computational Biology*, vol. 5, May 2009.

[59] R. Li, C. Yu, Y. Li, T.-W. Lam, S.-M. Yiu, K. Kristiansen, and J. Wang, "SOAP2: an improved ultrafast tool for short read alignment," *Bioinformatics*, vol. 25, Aug. 2009.

[60] F. Hach, F. Hormozdiari, C. Alkan, F. Hormozdiari, I. Birol, E. E. Eichler, and S. C. Sahinalp, "mrsFAST: a cache-oblivious algorithm for short-read mapping," *Nature Methods*, vol. 7, Aug. 2010.

[61] T. D. Wu and S. Nacu, "Fast and SNP-tolerant detection of complex variants and splicing in short reads," *Bioinformatics*, vol. 26, Apr. 2010.

[62] G. Rizk and D. Lavenier, "GASSST: global alignment short sequence search tool," *Bioinformatics*, vol. 26, Oct. 2010.

[63] M. David, M. Dzamba, D. Lister, L. Ilie, and M. Brudno, "SHRiMP2: Sensitive yet Practical Short Read Mapping," *Bioinformatics*, vol. 27, Apr. 2011.

[64] L. Egidi and G. Manzini, "Better spaced seeds using Quadratic Residues," *Journal of Computer and System Sciences*, vol. 79, Nov. 2013.

[65] B. Liu, D. Guan, M. Teng, and Y. Wang, "rHAT: fast alignment of noisy long reads with regional hashing," *Bioinformatics*, vol. 32, Jun. 2016.

[66] H. Li, "Minimap and miniasm: fast mapping and de novo assembly for noisy long sequences," *Bioinformatics*, vol. 32, Jul. 2016.

[67] R. Edgar, "Syncmers are more sensitive than minimizers for selecting conserved k-mers in biological sequences," *PeerJ*, vol. 9, Feb. 2021.

[68] K. Sahlin, "Effective sequence similarity detection with strobemers," *Genome Research*, vol. 31, Nov. 2021.

[69] K. Sahlin, "Strobealign: flexible seed size enables ultra-fast and accurate read alignment," *Genome Biology*, vol. 23, Dec. 2022.

[70] C. Firtina, J. Park, M. Alser, J. S. Kim, D. Cali, T. Shahroodi, N. Ghiasi, G. Singh, K. Kanellopoulos, C. Alkan, and O. Mutlu, "BLEND: a fast, memory-efficient and accurate mechanism to find fuzzy seed matches in genome analysis," *NAR Genomics and Bioinformatics*, vol. 5, Mar. 2023.

[71] M. Roberts, W. Hayes, B. R. Hunt, S. M. Mount, and J. A. Yorke, "Reducing storage requirements for biological sequence comparison," *Bioinformatics*, vol. 20, Dec. 2004.

[72] Y.-z. Zhang, A. Akdemir, G. Tremmel, S. Imoto, S. Miyano, T. Shibuya, and R. Yamaguchi, "Nanopore Basecalling from a Perspective of Instance Segmentation," *BMC bioinformatics*, 2020.

[73] R. Dias and A. Torkamani, "Artificial Intelligence in Clinical and Genomic Diagnostics," *Genome medicine*, vol. 11, 2019.

[74] S. L. Amarasinghe, S. Su, X. Dong, L. Zappia, M. E. Ritchie, and Q. Gouil, "Opportunities and Challenges in Long-Read Sequencing Data Analysis," *Genome biology*, vol. 21, 2020.

[75] F. J. Rang, W. P. Kloosterman, and J. de Ridder, "From Squiggle to Basepair: Computational Approaches for Improving Nanopore Sequencing Read Accuracy," *Genome Biology*, vol. 19, Jul 2018.

[76] Z. Wu, K. Hammad, R. Mittmann, S. Magierowski, E. Ghafar-Zadeh, and X. Zhong, "Fpga-based dna basecalling hardware acceleration," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2018.

[77] C. N. Ramachandra, A. Nag, R. Balasubramonion, G. Kalsi, K. Pillai, and S. Subramoney, "ONT-X: An FPGA approach to real-time portable genomic analysis," in *FCCM*, 2021.

[78] K. Hammad, Z. Wu, E. Ghafar-Zadeh, and S. Magierowski, "A scalable hardware accelerator for mobile DNA sequencing," *TVLSI*, 2021.

[79] Z. Wu, K. Hammad, A. Beyene, Y. Dawji, E. Ghafar-Zadeh, and S. Magierowski, "An FPGA implementation of a portable DNA sequencing device based on RISC-V," in *Newcas*, 2022.

[80] Z. Wu, K. Hammad, E. Ghafar-Zadeh, and S. Magierowski, "FPGA-accelerated 3rd generation DNA sequencing," *TBCS*, 2020.

[81] Q. Lou, S. C. Janga, and L. Jiang, "Helix: Algorithm/Architecture Co-design for Accelerating Nanopore Genome Base-calling," in *Proceedings of the ACM International Conference on Parallel Architectures and Compilation Techniques*, 2020.

[82] Q. Lou and L. Jiang, "Brawl: A spintronics-based portable basecalling-in-memory architecture for nanopore genome sequencing," *CAL*, 2018.

[83] B. Zoph and Q. V. Le, "Neural Architecture Search with Reinforcement Learning," *arXiv preprint arXiv:1611.01578*, 2016.

[84] C. Buciluă, R. Caruana, and A. Niculescu-Mizil, "Model Compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006.

[85] Y. LeCun, J. Denker, and S. Solla, "Optimal Brain Damage," *Advances in neural information processing systems*, vol. 2, 1989.

[86] Oxford Nanopore Technologies, "Dorado, https://github.com/nanoporetech/dorado."

[87] R. M. Gray and D. L. Neuhoff, "Quantization," *IEEE transactions on information theory*, vol. 44, 1998.

[88] 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, Jan. 2022.

[89] M. Alser, J. Rotman, D. Deshpande, K. Taraszka, H. Shi, P. I. Baykal, H. T. Yang, V. Xue, S. Knyazev, B. D. Singer, B. Balliu, D. Koslicki, P. Skums, A. Zelikovsky, C. Alkan, O. Mutlu, and S. Mangul, "Technology dictates algorithms: recent developments in read alignment," *Genome Biology*, vol. 22, Aug. 2021.

[90] H. Xin, D. Lee, F. Hormozdiari, S. Yedkar, O. Mutlu, and C. Alkan, "Accelerating read mapping with fasthash," *BMC Genomics*, 2013.

[91] H. Xin, J. Greth, J. Emmons, G. Pekhimenko, C. Kingsford, C. Alkan, and O. Mutlu, "Shifted Hamming Distance: A fast and accurate simd-friendly filter to accelerate alignment verification in read mapping," *Bioinformatics*, 2015.

[92] M. Alser, H. Hassan, H. Xin, O. Ergin, O. Mutlu, and C. Alkan, "GateKeeper: A new hardware architecture for accelerating pre-alignment in DNA short read mapping," *Bioinformatics*, 2017.

[93] J. S. Kim, D. Senol Cali, H. Xin, D. Lee, S. Ghose, M. Alser, H. Hassan, O. Ergin, C. Alkan, and O. Mutlu, "GRIM-Filter: Fast seed location filtering in DNA read mapping using processing-in-memory technologies," *BMC Genomics*, 2018.

[94] R. Kaplan, L. Yavits, and R. Ginosar, "RASSA: Resistive pre-alignment accelerator for approximate DNA long read mapping," *IEEE Micro*, 2018.

[95] M. Alser, H. Hassan, A. Kumar, O. Mutlu, and C. Alkan, "Shouji: A fast and efficient pre-alignment filter for sequence alignment," *Bioinformatics*, 2019.

[96] 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*, 2020.

[97] G. Singh, M. Alser, D. Senol Cali, D. Diamantopoulos, J. Gómez-Luna, H. Corporaal, and O. Mutlu, "FPGA-based near-memory acceleration of modern data-intensive applications," *IEEE Micro*, 2021.

[98] M. Alser, O. Mutlu, and C. Alkan, "MAGNET: Understanding and improving the accuracy of genome pre-Alignment filtering," *arXiv*, 2017.

[99] Z. Bingöl, M. Alser, O. Mutlu, O. Ozturk, and C. Alkan, "GateKeeper-GPU: Fast and accurate pre-alignment filtering in short read mapping," in *IPDPSW*. IEEE, 2021.

[100] M. Khalifa, R. Ben-Hur, R. Ronen, O. Leitersdorf, L. Yavits, and S. Kvatinsky, "FiltPIM: In-memory filter for DNA sequencing," in *ICECS*, 2021.

[101] N. Mansouri Ghiasi, J. Park, H. Mustafa, J. Kim, A. Olgun, A. Gollwitzer, D. Senol Cali, C. Firtina, H. Mao, N. Almadhoun Alserr, R. Ausavarungnirun, N. Vijaykumar, M. Alser, and O. Mutlu, "GenStore: A high-performance in-storage processing system for genome sequence analysis," in *ASPLOS*, 2022.

[102] T. Shahroodi, M. Zahedi, C. Firtina, M. Alser, S. Wong, O. Mutlu, and S. Hamdioui, "Demeter: A fast and energy-efficient food profiler using hyperdimensional computing in memory," *IEEE Access*, 2022.

[103] 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 *FCCM*, 2019.

[104] H. Sadasivan, M. Maric, E. Dawson, V. Iyer, J. Israeli, and S. Narayanasamy, "Accelerating Minimap2 for accurate long read alignment on GPUs," *bioRxiv*, 2022.

[105] Y. Chen, B. Schmidt, and D. L. Maskell, "A hybrid short read mapping accelerator," *BMC Bioinformatics*, 2013.

[106] S. K. Khatamifard, Z. Chowdhury, N. Pande, M. Razaviyayn, C. Kim, and U. R. Karpuzcu, "Read mapping near non-volatile memory," *arXiv*, 2017.

[107] Y. Turakhia, G. Bejerano, and W. J. Dally, "Darwin: A genomics co-processor provides up to 15,000x acceleration on long read assembly," in *ASPLOS*, 2018.

[108] S. D. Goenka, Y. Turakhia, B. Paten, and M. Horowitz, "SegAlign: A scalable GPU-based whole genome aligner," in *SC*, 2020.

[109] A. Nag, C. N. Ramachandra, R. Balasubramonian, R. Stutsman, E. Giacomin, H. Kambalasubramanyam, and P.-E. Gaillardon, "GenCache: Leveraging in-Cache operators for efficient sequence alignment," in *MICRO*, 2019.

[110] Q. Aguado-Puig, S. Marco-Sola, J. C. Moure, D. Castells-Rufas, L. Alvarez, A. Espinosa, and M. Moreto, "Accelerating edit-distance sequence alignment on GPU using the wavefront algorithm," *IEEE Access*, 2022.

bioRxiv preprint doi: https://doi.org/10.1101/2023.01.22.525080; this version posted January 23, 2023. The copyright holder for this preprint (which was not certified by peer review) is the author/funder, who has granted bioRxiv a license to display the preprint in perpetuity. It is made available under aCC-BY-NC-ND 4.0 International license.

[111] Q. Aguado-Puig, S. Marco-Sola, J. C. Moure, C. Matzoros, D. Castells-Rufas, A. Espinosa, and M. Moreto, "WFA-GPU: Gap-affine pairwise alignment using GPUs," *bioRxiv*, 2022.

[112] A. Haghi, S. Marco-Sola, L. Alvarez, D. Diamantopoulos, C. Hagleitner, and M. Moreto, "An FPGA accelerator of the wavefront algorithm for genomics pairwise alignment," in *FPL*, 2021.

[113] D. Senol Cali, G. Kalsi, Z. Bingöl, L. Subramanian, C. Firtina, J. Kim, R. Ausavarungnirun, M. Alser, A. Nori, J. Luna *et al.*, "GenASM: A high-performance, low-power approximate string matching acceleration framework for genome sequence analysis," in *MICRO*, 2020.

[114] J. Lindegger, D. S. Cali, M. Alser, J. Gómez-Luna, and O. Mutlu, "Algorithmic improvement and GPU acceleration of the GenASM algorithm," *arXiv*, 2022.

[115] J. Lindegger, D. S. Cali, M. Alser, J. Gómez-Luna, N. M. Ghiasi, and O. Mutlu, "Scrooge: A fast and memory-frugal genomic sequence aligner for CPUs, GPUs, and ASICs," *arXiv*, 2022.

[116] D. Senol Cali, K. Kanellopoulos, J. Lindegger, Z. Bingöl, G. S. Kalsi, Z. Zuo, C. Firtina, M. B. Cavlak, J. Kim, N. M. Ghiasi, G. Singh, J. Gómez-Luna, N. A. Alserr, M. Alser, S. Subramoney, C. Alkan, S. Ghose, and O. Mutlu, "SeGraM: A universal hardware accelerator for genomic sequence-to-graph and sequence-to-sequence mapping," in *ISCA*, 2022.

[117] D. Fujiki, A. Subramaniyan, T. Zhang, Y. Zeng, R. Das, D. Blaauw, and S. Narayanasamy, "GenAx: A genome sequencing accelerator," in *ISCA*, 2018.

[118] A. Madhavan, T. Sherwood, and D. Strukov, "Race Logic: A hardware acceleration for dynamic programming algorithms," *CAN*, 2014.

[119] H. Cheng, Y. Zhang, and Y. Xu, "Bitmapper2: A GPU-accelerated all-mapper based on the sparse Q-gram index," *TCBB*, 2018.

[120] E. J. Houtgast, V.-M. Sima, K. Bertels, and Z. Al-Ars, "Hardware acceleration of BWA-MEM genomic short read mapping for longer read lengths," *Computational Biology and Chemistry*, 2018.

[121] E. J. Houtgast, V. Sima, K. Bertels, and Z. AlArs, "An efficient GPU-accelerated implementation of genomic short read mapping with BWA-MEM," *CAN*, 2017.

[122] A. Zeni, G. Guidi, M. Ellis, N. Ding, M. D. Santambrogio, S. Hofmeyr, A. Buluç, L. Oliker, and K. Yelick, "Logan: High-performance GPU-based X-drop long-read alignment," in *IPDPS*, 2020.

[123] N. Ahmed, J. Lévy, S. Ren, H. Mushtaq, K. Bertels, and Z. Al-Ars, "GASAL2: A GPU accelerated sequence alignment library for high-throughput NGS data," *BMC Bioinformatics*, 2019.

[124] T. Nishimura, J. L. Bordim, Y. Ito, and K. Nakano, "Accelerating the Smith-waterman algorithm using bitwise parallel bulk computation technique on GPU," in *IPDPSW*, 2017.

[125] E. F. de Oliveira Sandes, G. Miranda, X. Martorell, E. Ayguade, G. Teodoro, and A. C. M. Melo, "CUDAlign 4.0: Incremental speculative traceback for exact chromosome-wide alignment in GPU clusters," *TPDS*, 2016.

[126] Y. Liu and B. Schmidt, "GSWABE: Faster GPU-accelerated sequence alignment with optimal alignment retrieval for short DNA sequences," *Concurrency and Computation: Practice and Experience*, 2015.

[127] Y. Liu, A. Wirawan, and B. Schmidt, "CUDASW++ 3.0: Accelerating Smith-Waterman protein database search by coupling CPU and GPU SIMD instructions," *BMC Bioinformatics*, 2013.

[128] Y. Liu, D. L. Maskell, and B. Schmidt, "CUDASW++: Optimizing Smith-Waterman sequence database searches for CUDA-enabled graphics processing units," *BMC Research Notes*, 2009.

[129] Y. Liu, B. Schmidt, and D. L. Maskell, "CUDASW++ 2.0: Enhanced Smith-Waterman protein database search on CUDA-enabled GPUs based on SIMT and virtualized SIMD abstractions," *BMC Research Notes*, 2010.

[130] R. Wilton, T. Budavari, B. Langmead, S. J. Wheelan, S. L. Salzberg, and A. S. Szalay, "Arioc: High-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space," *PeerJ*, 2015.

[131] A. Goyal, H. J. Kwon, K. Lee, R. Garg, S. Y. Yun, Y. H. Kim, S. Lee, and M. S. Lee, "Ultra-fast next generation human genome sequencing data processing using DRAGEN Bio-IT processor for precision medicine," *OJGen*, 2017.

[132] Y.-T. Chen, J. Cong, Z. Fang, J. Lei, and P. Wei, "When Spark Meets FPGAs: A case study for next-generation DNA sequencing acceleration," in *HotCloud*, 2016.

[133] A. F. Laguna, H. Gamaarachchi, X. Yin, M. Niemier, S. Parameswaran, and X. S. Hu, "Seed-and-Vote based in-memory accelerator for DNA read mapping," in *ICCAD*, 2020.

[134] P. Chen, C. Wang, X. Li, and X. Zhou, "Accelerating the next generation long read mapping with the FPGA-based system," *TCBB*, 2014.

[135] Y.-L. Chen, B.-Y. Chang, C.-H. Yang, and T.-D. Chiueh, "A high-throughput FPGA accelerator for short-read mapping of the whole human genome," *TPDS*, 2021.

[136] D. Fujiki, S. Wu, N. Ozog, K. Goliya, D. Blaauw, S. Narayanasamy, and R. Das, "SeedEx: A genome sequencing accelerator for optimal alignments in subminimal space," in *MICRO*, 2020.

[137] S. S. Banerjee, M. El-Hadedy, J. B. Lim, Z. T. Kalbarczyk, D. Chen, S. S. Lumetta, and R. K. Iyer, "ASAP: Accelerated short-read alignment on programmable hardware," *TC*, 2019.

[138] X. Fei, Z. Dan, L. Lina, M. Xin, and Z. Chunlei, "FPGASW: Accelerating large-scale Smith–Waterman sequence alignment application with backtracking on FPGA linear systolic array," *Interdisciplinary Sciences: Computational Life Sciences*, 2018.

[139] H. M. Waidyasooriya and M. Hariyama, "Hardware-acceleration of short-read alignment based on the Burrows-wheeler transform," *TPDS*, 2015.

[140] X.-Q. Li, G.-M. Tan, and N.-H. Sun, "PIM-Align: A processing-in-memory architecture for FM-Index search algorithm," *JCST*, 2021.

[141] Y.-T. Chen, J. Cong, J. Lei, and P. Wei, "A novel high-throughput acceleration engine for read alignment," in *FCCM*, 2015.

[142] E. Rucci, C. Garcia, G. Botella, A. De Giusti, M. Naiouf, and M. Prieto-Matias, "SWIFOLD: Smith-Waterman implementation on FPGA with OpenCL for long DNA sequences," *BMC Systems Biology*, 2018.

[143] S. Diab, A. Nassereldine, M. Alser, J. Gómez-Luna, O. Mutlu, and I. E. Hajj, "A framework for high-throughput sequence alignment using real processing-in-memory systems," *arXiv*, 2022.

[144] F. Zokaee, M. Zhang, and L. Jiang, "FindeR: Accelerating FM-index-based exact pattern matching in genomic sequences through ReRAM technology," in *PACT*, 2019.

[145] S. Angizi, W. Zhang, and D. Fan, "Exploring DNA alignment-in-memory leveraging emerging SOT-MRAM," in *GLSVLSI*, 2020.

[146] S. Diab, A. Nassereldine, M. Alser, J. G. Luna, O. Mutlu, and I. E. Hajj, "High-throughput pairwise alignment with the wavefront algorithm using processing-in-memory," *arXiv*, 2022.

[147] W. Huangfu, S. Li, X. Hu, and Y. Xie, "RADAR: A 3D-ReRAM based DNA alignment accelerator architecture," in *DAC*, 2018.

[148] Z. I. Chowdhury, M. Zabihi, S. K. Khatamifard, Z. Zhao, S. Resch, M. Razaviyayn, J.-P. Wang, S. S. Sapatnekar, and U. R. Karpuzcu, "A DNA read alignment accelerator based on computational RAM," *JXCDC*, 2020.

[149] L. Li, J. Lin, and Z. Wang, "PipeBSW: A two-stage pipeline structure for banded Smith-Waterman algorithm on FPGA," in *ISVLSI*, 2021.

[150] L. Wu, D. Bruns-Smith, F. A. Nothaft, Q. Huang, S. Karandikar, J. Le, A. Lin, H. Mao, B. Sweeney, K. Asanović *et al.*, "FPGA accelerated indel realignment in the cloud," in *HPCA*, 2019.

[151] Y. Yan, N. Chaturvedi, and R. Appuswamy, "Accel-Align: a fast sequence mapper and aligner based on the seed–embed–extend method," *BMC Bioinformatics*, 2021.

[152] J. Daily, "Parasail: SIMD C library for global, semi-global, and local pairwise sequence alignments," *BMC Bioinformatics*, 2016.

[153] S. Kalikar, C. Jain, M. Vasimuddin, and S. Misra, "Accelerating minimap2 for long-read sequencing applications on modern CPUs," *Nature Computational Science*, 2022.

[154] S. Marco-Sola, J. C. Moure, M. Moreto, and A. Espinosa, "Fast gap-affine pairwise alignment using the wavefront algorithm," *Bioinformatics*, 2021.

[155] R. Kaplan, L. Yavits, R. Ginosar, and U. Weiser, "A resistive CAM processing-in-storage architecture for DNA sequence alignment," *IEEE Micro*, 2017.

[156] S. K. Khatamifard, Z. Chowdhury, N. Pande, M. Razaviyayn, C. H. Kim, and U. R. Karpuzcu, "GeNVoM: Read mapping near non-volatile memory," *TCBB*, 2021.

[157] F. Chen, L. Song, Y. Chen *et al.*, "PARC: A processing-in-CAM architecture for genomic long read pairwise alignment using ReRAM," in *ASP-DAC*, 2020.

[158] S. Gupta, M. Imani, B. Khaleghi, V. Kumar, and T. Rosing, "RAPID: A reRAM processing in-memory architecture for DNA sequence alignment," in *ISLPED*, 2019.

[159] F. Zokaee, H. R. Zarandi, and L. Jiang, "AligneR: A process-in-Memory architecture for short read alignment in ReRAMs," *CAL*, 2018.

[160] J. M. Eizenga and B. Paten, "Improving the time and space complexity of the WFA algorithm and generalizing its scoring," *bioRxiv*, 2022.

[161] C. Firtina, K. Pillai, G. S. Kalsi, B. Suresh, D. S. Cali, J. Kim, T. Shahroodi, M. B. Cavlak, J. Lindegger, M. Alser, J. G. Luna, S. Subramoney, and O. Mutlu, "Aphmm: Accelerating profile hidden markov models for fast and energy-efficient genome analysis," *arXiv*, Jul. 2022.

[162] S. Marco-Sola, J. M. Eizenga, A. Guarracino, B. Paten, E. Garrison, and M. Moreto, "Optimal gap-affine alignment in O(s) space," *bioRxiv*, 2022.