Enhancing Sequence Alignment Efficiency Through Concurrent Utilization of Multiple Arm Processors in a Sequential Processing Framework

Yunzi Dai, Liwei Liu, Zhuochen Yang, and Shaolong Chen(✉)

Jiangxi Provincial Engineering Research Center of Blockchain Data Security and Governance, Jiangxi Normal University, Nanchang 330022, JX, China
schen@jxnu.edu.cn

Abstract. High performance computing (HPC) solutions traditionally meet the intense computational demands inherent in genome processing. Components of genome processing have been implemented on GPUs, FPGAs, and ASICs. However, they are primarily used as coprocessors for processor servers rather than as independent running systems. The embedded systems with Arm processors have garnered increasing attention over the years. This study introduces an intelligent technique for short read genome alignment, leveraging advanced Arm-based processors. Our novel system integrates a sequential workflow of Arm-based processors, an intelligent mechanism for partitioning the reference genome, a timer-based system for detecting alignment, and a technique for optimizing memory. This results in accelerated alignment with efficient resource utilization and minimization of unproductive searches. The main focus of this study is the implementation of a workflow that reduces memory access and per-processor footprint, providing a revolutionary approach to aligning short read genomes. Through testing millions of simulated sequences, our system significantly improved both alignment speed and result accuracy.

Keywords: HPC · Embedded system · Arm processor · Sequence alignment · Variant analysis

1 Introduction

With the emergence of next-generation sequencing (NGS) technologies, the amount of genome sequence data has been increasing at an exponential rate. This boom brings several challenges. NGS generates massive amounts of data, requiring extensive storage solutions. Storing this data efficiently and cost-effectively becomes a significant issue. The huge amount of NGS data demands high computational power and sophisticated algorithms to process and interpret. Existing computational infrastructure and tools may struggle to keep up.

Y. Cao and X. Shao (Eds.): DIONE 2023, LNICST 515, pp. 216–228, 2025.
https://doi.org/10.1007/978-3-031-80713-8_16

However, sequence alignment is a computationally intensive process. It involves comparing a sequence of nucleotides (DNA or RNA) or amino acids (proteins) to a reference sequence in order to determine the degree of similarity. This is a key step in many bioinformatics analyses, including genome assembly, variant detection, phylogenetic analysis, and many others. Due to these factors, such as the size of the dataset, complexity of the algorithm, reference genome size, and quality of the sequencing data, considerable effort has been devoted to developing more efficient alignment algorithms and software, as well as using high-performance computing, cloud computing, and parallel processing to address the computational challenges associated with sequence alignment.

Currently, genome processing largely depends on expansive computational servers or cloud-based solutions, creating reliance on these massive systems for complex genomic tasks. This dependency limits accessibility and raises concerns about data privacy and efficiency. A platform, powered by embedded processors, is engineered to handle the intricate task of genome processing but is small enough to be integrated into a portable device. This would democratize genomic data analysis, making it accessible to researchers and healthcare professionals in any setting, not just those with access to large-scale computational resources. With such innovation, we can foresee the development of portable genome processing devices. These would not only transform the field of genomic research but also have profound impacts on healthcare by enabling real-time, onsite genomic data analysis.

In order to realize the potential of portable genome computation devices, it is crucial to transition the computational load to smaller, Arm-based processors. However, this shift is not without its challenges. The nature of genome computation, with its enormous data sizes and complex analytical processes, presents significant hurdles when attempting to transition to a more compact processing environment. These embedded processors must be able to handle not only the vast amounts of genetic data but also execute the intricate algorithms necessary for genome sequencing and analysis.

In this paper, we introduce a pioneering methodology for short read genome alignment leveraging embedded processors through a sequential workflow architecture. We deploy a range of innovative strategies to optimize alignment speed, computational efficiency, and memory usage. Our approach begins with a highly structured framework that facilitates simultaneous task execution, reduces redundancy, and accelerates the alignment process. The use of multiple Arm-based processor systems makes this strategy possible. Besides, we use an intelligent partitioning approach to manage the reference genome. By indexing and storing smaller segments, our method efficiently uses resources and enhances alignment speed by limiting the range of data needed to be searched. We further include a timer-based alignment failure detection system. This mechanism prevents lengthy, unproductive searches by terminating a search that exceeds a designated duration without finding a match and moving on to the next potential match. Additionally, our approach adopts a memory optimization strategy aimed at enhancing data retrieval speed during the alignment process. This is particularly beneficial for embedded processors with limited memory capacity. Our methodology also considers the incorporation of multiple sequence aligner techniques, which could enhance the learning ability of system from each alignment process, thus improving accuracy and speed over time.

The major contribution of this research is the introduction of a sequential frame structure that partitions the genome reference to minimize memory accesses and reduce the memory footprint required by each small processor. We showcase the results of aligning various millions of simulated sequence reads to the full human genome, demonstrating that our solution markedly speeds up the short read alignment process. Therefore, our innovative architecture facilitates genomic sequence alignment, enabling comprehensive genome-wide alignment.

The structure of the paper is as follows: In Sect. 2, we delve into the context by discussing sequence alignment and general HPC hardware. Section 3 introduces our proposed methodologies in details. Besides, Sect. 4 is where we present and evaluate our results in terms of alignment performance and result accuracy. In addition, Sect. 5 discusses the existent research in this field. It examines studies and literature pertinent to our work. Finally, Sect. 6 concludes the paper and makes suggestions for future research directions.

2 Background

2.1 Sequence Alignment

Sequence alignment is a molecular biology method used to identify similarities between DNA, RNA, or protein sequences. By comparing sequences from different organisms or different genes within the same organism, researchers can infer structural, functional, and evolutionary relationships. Alignment operates on the principle that regions of certain sequences can be conserved across species due to evolutionary pressures, and these conserved sequences often correspond to important biological functions.

In sequence alignment, the characters (nucleotides for DNA and RNA and amino acids for proteins) in different sequences that are thought to have evolved from the same ancestor are put in columns that are next to each other. These are called homologous positions.

There are two mainstream types of sequence alignment:

1. Global alignment, which attempts to align every residue in every sequence. An example of a global alignment algorithm is the Needleman-Wunsch algorithm.
2. Local alignment, which attempts to align subregions of sequences that are most similar, the Smith-Waterman algorithm is an example of this.

Table 1 presents the fundamental categorization of these widely used sequence aligners and algorithms. Certain aligners, such as BLAST, have the capability to execute both local and global alignment procedures, contingent upon the specific configurations and application scenarios. The categorization of aligners into short-read and long-read aligners is mostly determined by the most suitable input data. However, it is worth noting that certain tools have the capability to accommodate a wider spectrum of sequence lengths in certain instances. It is advisable to refer to the particular documentation provided by the aligner in order to obtain the most precise and comprehensive information.

Table 1. The popular sequence aligners and alignment algorithms

Aligner Name	Type	Year	Sequence Type
BLAST	Local and global alignment	1990	Nucleotide, Protein
Bowtie	Short-read alignment	2009	Nucleotide
BWA	Short-read alignment	2009	Nucleotide
minimap2	Long-read alignment	2016	Nucleotide
MAFFT	Multiple sequence alignment	1997	Nucleotide, Protein
Clustal Omega	Multiple sequence alignment	2011	Nucleotide, Protein
Muscle	Multiple sequence alignment	2004	Nucleotide, Protein

2.2 General Hardware Accelerators in HPC

In HPC, a hardware accelerator is a component that can perform certain tasks more efficiently than a general-purpose central processing unit (CPU). Accelerators are designed to perform specific tasks with great efficiency, thus freeing up the CPU for other tasks and overall increasing the performance of the system. They can speed up data processing, reduce energy consumption, and tackle complex algorithms.

Table 2. Hardware comparison

Accelerators	Pros	Cons
GPU	1. high degree of parallelism 2. robust software ecosystems	1. not all tasks can be effectively parallelized 2. power consumption can be high
FPGA	1. can be customized for specific tasks 2. offering potentially very high performance 3. lower power consumption than GPUs	1. require specialized knowledge to program 2. not as flexible as GPUs once programmed for a specific task 3. can be expensive
ASIC	1. extremely efficient at the task they are designed for 2. low power consumption compared to GPUs and FPGAs	1. very high design and manufacturing costs 2. lack flexibility once designed 3. cannot be reprogrammed for other tasks

Some examples of hardware accelerators are Graphics Processing Units (GPUs), Field-Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs).

1. GPUs were originally designed for fast processing of graphics data, but their architecture, which allows many tasks to be carried out simultaneously, has proved advantageous for other types of computations, including sequence alignment.

2. FPGAs are programmable silicon chips that can be tailored to execute specific tasks extremely efficiently. For sequence alignment, an FPGA can be programmed to implement a specific alignment algorithm directly in hardware, potentially resulting in major speed improvements.
3. ASICs are custom chips designed to do a particular job very efficiently. ASICs for sequence alignment could, in theory, offer superior performance and efficiency, but their high development cost and lack of flexibility have so far limited their use in this field.

When it comes to sequence alignment, hardware accelerators can significantly speed up the process. Due to the vast amount of data involved in genomic sequences, the process of alignment can be computationally demanding. Hardware accelerators can parallelize these operations, providing a much-needed speed boost. Tools like BarraCUDA (for GPUs) or SOAP3 (which utilizes both CPUs and GPUs) have been developed to leverage the computational power of these accelerators in sequence alignment tasks.

The following is a comparison table for these three types of hardware accelerators in Table 2. For sequence alignment, which hardware accelerator is suitable for sequence alignment that relies on specific factors such as the size and the nature of the datasets, available resources, and the particular alignment algorithm being used.

2.3 Arm-Based Processor

Arm-based processors are known for their power efficiency and lower heat generation, which originally made them an excellent choice for mobile devices like smartphones and tablets. However, over the years, Arm architecture has evolved significantly to offer a wide range of solutions, from single-core microcontrollers to multicore processors suitable for servers and data centers.

HPC, which requires significant computational power, traditionally relied on more power-hungry architectures such as those provided by Intel and AMD. However, the landscape has started to shift. Arm-based processors, with their increased performance capabilities and power efficiency, have begun to make their way into the HPC environment.

While traditionally, sequence alignment tasks have been carried out using more conventional processors, like those based on Intel or AMD architectures, Arm-based processors have been gaining traction due to their power efficiency and the increasing performance they offer. This process can be computationally intensive, especially for large genomes or large quantities of data. High-performance computing systems, including those with Arm-based processors, can be employed to speed up this process and handle larger datasets.

However, it is worth noting that the specific software used for sequence alignment needs to be compatible with the Arm architecture. Given the increasing popularity of Arm-based systems in HPC and server environments, many popular bioinformatics tools have been or are being ported to run on Arm architectures.

We applied an Arm-based processor in our following experiments. A simple table summarizing some key characteristics of the Arm Cortex-A53 CPU we applied in the experiments is shown in Table 3.

Table 3. Key characteristics of the Arm Cortex-A53 processor

Hardware Parameter	Description
Architecture	Arm v8-A (AArch64)
Clock Rate	Up to 1.8 GHz
Cores	4 cores
Memory	1 GB DDR3
L1 Cache	8 KB/16 KB/32 KB/64 KB
L2 Cache	Up to 1 MB
Process	Down to 16 nm
Memory	40-bit physical addressing

3 The Proposed Methodology

Our research introduces a series of innovative strategies aimed at optimizing genome alignment processes. The proposed methodology, based on an Arm-based processor, employs four pivotal techniques aimed at curtailing the duration of short read alignment.

The first notable strategy is the introduction of a workflow pipeline based on the Arm processor system. This workflow provides a well-structured fashion that minimizes redundancy by enabling the simultaneous execution of multiple tasks. This simultaneous processing capacity significantly accelerates the alignment process, enhancing alignment efficiency and computation throughput.

The second strategy revolves around an intelligent partitioning approach implemented by our proposed method. It involves dividing the reference genome into smaller, more manageable segments. These sections are indexed and stored for easy retrieval. This technique optimizes resource utilization and significantly boosts the speed of alignment by narrowing the range of data to be searched during the alignment process.

Incorporating the third technique, the proposed method establishes a timer-based alignment failure detection system. This mechanism aims to stop wasteful, prolonged searches brought on by genome partitioning. When a search exceeds a designated duration without successfully identifying a match, the timer mechanism is activated, allowing the system to terminate the fruitless search and advance to the next potential match.

Furthermore, the proposed method adopts a memory optimization strategy as the fourth technique. This strategy aims to optimize the use of memory, enhancing data retrieval speed during the alignment process. This becomes particularly advantageous in the context of embedded processors with limited memory capacity.

To extend our methods, we are also exploring the integration of multiple sequence aligner techniques. Such an approach could enable the system to learn from each alignment process, improving accuracy and speed over time and making comparisons among various aligners.

Additionally, we are considering a distributed computing approach based on Arm processors, which would allow multiple processor systems to collaborate on alignment

tasks simultaneously, potentially leading to significant speed improvements. The effectiveness and efficiency of the proposed method in future applications can be further improved by these complementary strategies.

The following pseudocodes illustrate the procedures of the proposed methodologies we discuss above, which consist of pipelined workflow, intelligent partitioning, alignment timer, and memory optimization.

```
# Pseudocode for Pipelined Workflow
While (tasks remain in queue) do
    For each task in queue do
        If (task can be executed on Arm processor) then
            Dispatch task to Arm processor
            Remove task from queue
        End if
    End for
End while
```

```
# Pseudocode for Alignment Timer
While (searching for match) do
    Start Timer
    If (match found) then
        Stop Timer
        Break
    Else if (Timer exceeds threshold) then
        Stop Timer
        Terminate Search
        Move to next potential match
    End if
End while
```

```
# Pseudocode for Memory Optimization
If data in cache then
    Return cache[data]
Else Load_data_from_memory(data) = data
    Store_in_cache(data, cache)
    Return data
End if
```

```
# Pseudocode for Intelligent Partitioning
Partition Reference Genome into manageable segments
For each segment do
    Index segment
    Store segment in memory
End for
```

4 The Results and the Evaluations of the Experiments

To comprehensively assess the performance enhancement and alignment accuracy of the proposed methodology, a total of one million sequence reads with random length were strategically aligned against the entirety of the human genome.

Table 4. Key configurations in the experiments

Configurations	Value or Description
Total number of sequence reads	1,000,000
Length of each sequence read	Random
Number of distinct groups (N)	1, 2, 4, 8, 16, 32
Number of processors in the pipeline	Determines the value of N
Genome partitioning	Human genome divided into N segments with overlapping regions
Overlapping region size	Fixed-size base pair

In preparation for the alignment process, the synthetic reads were partitioned into N distinct groups, each group corresponding to a separate file. The number of processors in the pipeline determines the value of N, with N = 1, 2, 4, 8, 16, and 32 being potential configurations. Simultaneously, the full human genome reference was meticulously divided into N segments, each incorporating an overlapping region of a fixed-size base pair. This overlapping region is incorporated to ensure the comprehensiveness of the alignment, particularly at the boundaries of each partitioned segment.

After the partitioning phase, each section of the genome underwent an individual pre-processing and indexing technique. This step was conducted offline in order to preserve computer resources. The utilization of this indexed data structure is of utmost importance in facilitating following effective searching and alignment procedures. It is important to acknowledge that the utilization of this partitioning strategy, which aims to save resources, is a fundamental aspect of the suggested methodology. This approach successfully improves the performance of alignment by lowering both the computational workload and the memory usage. Key configurations in the experiments is shown in Table 4.

4.1 Evaluation of Performance

To conduct a rigors comparative analysis of the performance of the proposed methodology, five distinct system configurations were deployed, each employing a different number of processors: 2, 4, 8, 16, 32, respectively.

In the first experiment, the baseline point was measured with a single processor only. This configuration can be denoted as baseline. Followed by the experiment referred to as a linear partition. The processors were arranged in a linear configuration, with each processor being assigned an equally-sized partition of the reference genome. All synthetic reads were initially introduced to the first processor. After excluding the reads that were precisely aligned, all the remaining reads were passed on sequentially to the following processor in the pipeline. The third experimental condition entailed an increasing partition, in which the reference genome was partitioned into segments of progressively increasing size. This adjustment was implemented with the objective of achieving a balanced computational load distribution between the processors situated at the front and back ends of the pipeline. The fourth experiment involved the deployment of sequence alignment Arm-based processor without a timer mechanism, which is denoted as no timer alignment. The aim of this experiment was to evaluate the performance and efficiency of the method without accelerating alignment fault detection through a timer mechanism. In the final experiment, the alignment approach was implemented with the inclusion of a timer mechanism, which is denoted as timer alignment. The comparison of the outcomes from this experiment and the previous one allows for the analysis of the extent to which the inclusion of a timer mechanism contributes to the enhancement of performance and efficiency of the proposed methodology.

Figure 1 provides a comparative analysis of the computational speed for the five experiments we conducted. In these tests, we had to estimate the time for a single processor since its memory capacity, 1 GB only for 4 cores in a single processor, was not sufficient to accommodate the entirety of the human genome, about 3 GB. Therefore, we estimated the time-cost in these cases by using the pipeline workflow. By contrast,

when we employed 32 processors using the Arm-based system, we achieved a significant increase in efficiency. If we permited timer, the process speed was augmented by 6.8X compared to the single-processor baseline scenario. Even when we did not allow timer, there was still an impressive 4.2X speed increase. The implementation of an increasing partition further boosted the performance of system. In fact, the improvement was a speedup of 2.8X the original speed. However, when we tried to utilize the linear partition with an equal reference distribution across 32 processors, we observed no remarkable increase in efficiency, only a 1.7X speedup. The reason behind this lack of improvement was the imbalance of many processors in the system. The workload was not distributed in a manner that kept all processors sufficiently engaged, leading to underutilization of the full capacity of system.

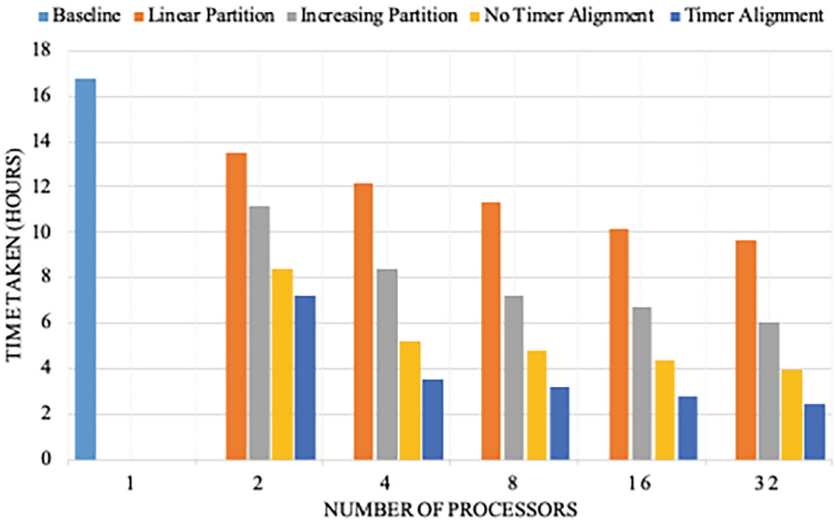

Fig. 1. Performance analysis with various processors in alignment

In conclusion, employing multiple processors Arm-based system with optimized workload distribution and timer utilization significantly increased computational speed and efficiency, highlighting the importance of strategic partitioning and workload management in large-scale data processing tasks.

4.2 Evaluation of Result Accuracy

In order to authenticate the accuracy of the resulting output code modification, a detailed comparison was conducted between the individual alignment scores generated by the baseline test, which utilized the original sequence aligner running on an Intel CPU, and those produced by the Arm-based system incorporating a timer mechanism. The accuracy metrics are defined in several ways according to special targets. In this research, we suppose it refers to the percentage of aligned sequences relative to the total number of sequences.

Table 5 shows the accuracy of four different sequence alignment methods on two different hardware platforms (Intel and Arm). The accuracy they achieve is generally almost identical. BWA-ALN has an accuracy of 92.6% on Intel and 91.6% on Arm. The accuracy difference between the two platforms for BWA-ALN is not substantial. The accuracy of BWA-MEM on Intel is 93.4%, while on Arm, it is 92.3%. Similar to BWA-ALN, BWA-MEM also shows a minimal difference in accuracy across the two platforms. Alignment with the timer on Intel achieves an accuracy of 96.2%, whereas on Arm, it has an accuracy of 93.3%. This has the highest accuracy among the four methods on the Intel platform but performs less well on the Arm relative to Intel. Alignment without the timer method has an accuracy of 94.8% on Intel and 92.5% on Arm. It has relatively high accuracy on both platforms, though it is less accurate on Arm.

Table 5. Accuracy results with various aligners in different platforms

Method	Accuracy (%) in the Intel	Accuracy(%) in the Arm
BWA-ALN	92.6	91.6
BWA-MEM	93.4	92.3
BWA-Timer alignment	96.2	93.3
BWA-No timer alignment	94.8	92.5

From Table 5, we can see that all alignment methods have quite similar accuracy on the Intel CPU than on the Arm processor. This could be because these methods are better optimized for Intel hardware or that Intel hardware is more suited for these alignment operations. But an Arm-based system could give us a cheaper and more convenient way to utilize the calculation of sequence alignment in variant analysis. Additionally, alignment with the timer has the highest accuracy on both platforms, suggesting that the timer mechanism may aid in increasing alignment accuracy. However, for more precise conclusions, additional experiments about the test conditions and hardware configurations may be necessary.

5 The Related Work

Sequence alignment is a fundamental operation in bioinformatics. It is employed in many applications, such as genome assembly, gene prediction, phylogenetic analysis, and protein structure prediction. Despite its importance, sequence alignment is a computationally intense task, especially when dealing with large datasets common in current genomics era. As such, hardware accelerators, including GPUs, FPGAs, and ASICs, have been increasingly utilized to expedite sequence alignment procedures. This section summarizes recent advances in the application of hardware accelerators for sequence comparison, discussing their performance, advantages, and challenges.

GPUs can operate thousands of threads simultaneously, making them suitable for sequence alignment. Many GPU-implemented algorithms are faster than CPU-implemented ones. CUDAlign [1] is a GPU algorithm built using CUDA. Global alignment and GPU memory-fitting fragmentation allow it to handle enormous sequences.

One other GPU-based sequence alignment tool that uses BWA-MEM is BarraCUDA [2]. Ideal for aligning next-generation sequencing data. CUSHAW3 is a popular GPU-based sequence alignment tool [3]. CUSHAW3, a parallelized short-read aligner, aligns sequences using CUDA. It aligns single-end and paired-end reads and handles high-throughput next-generation sequencing data. The CUSHAW3 outperforms other top aligners in speed and accuracy. NVBIO [4], an open-source C++ library from NVIDIA, is another GPU-based utility. NVBIO offers accurate and approximate sequence alignment techniques for bioinformatics applications. Its versatile interface enables people add features simply.

In terms of research on GPUs, there has been a focus on optimizing the performance of sequence alignment on GPUs. This includes developing new algorithms that can better exploit the parallelism of GPUs as well as techniques for managing the limited memory capacity of these devices. For instance, some research has looked at using compressed data structures [5] to reduce the memory footprint of sequence alignment on GPUs.

FPGAs are reprogrammable silicon chips that can be customized to perform specific computational tasks. They offer a good balance between flexibility and performance, providing higher speedups than GPUs in some cases, although they require more specific knowledge to program. One of the notable FPGA-based sequence alignment tools is the OpenCL Library [6], which utilizes the OpenCL framework. It offers an interface similar to software programming for easier adoption by bioinformaticians. Another example of an FPGA-based tool is "FASTA-Stack" [7], a system designed for high-throughput sequence alignment. It is based on a reconfigurable and scalable FPGA-accelerated hardware stack that delivers up to 240 times faster speeds than the FASTA tool. Another FPGA-based sequence alignment tool is "CBESW" [8], a systolic array-based implementation of the Smith-Waterman algorithm on FPGA. It is specifically designed for comparing a small number of long sequences. The tool leverages the parallelism and high-speed arithmetic capabilities of FPGAs to provide high-performance sequence alignment.

In terms of research on FPGAs, several studies have focused on optimizing the design and implementation of sequence alignment algorithms on FPGAs. This includes work on improving the efficiency of memory usage, developing techniques for reducing the communication overhead between the FPGA and the host computer, and investigating new ways of exploiting the parallelism of FPGAs.

ASICs are chips designed for a specific application, providing the highest performance and energy efficiency but at the cost of flexibility. ASIC-based sequence aligners are less common due to their high development cost and lack of flexibility, but they are representative of the most advanced performance. The DRAGEN Bio-IT [9] platform is a notable example that utilizes ASIC technology. It offers ultra-rapid sequence alignment and variant calling, although its use is limited due to the high cost of the platform. The writers also talk about how they made the Smith-Waterman algorithm work better for their ASIC design and show that DRAGEN can do variant calling and sequence alignment a lot faster than CPU-based systems. Research [10] describes the design of a custom ASIC for accelerating the process of short read mapping, a key step in sequence alignment. The authors discuss the challenges of implementing the complex algorithms used in read mapping on an ASIC and present their solutions. Research [11] presents a

sequence alignment algorithm specifically designed for a fine-grained, many-core system, an architecture that can be implemented on an ASIC. The authors show that their algorithm provides both high performance and energy efficiency.

In terms of research in ASICs, there is ongoing work on developing more efficient and cost-effective ASIC designs for sequence alignment, as well as investigating new applications for this technology in genomics. With the ongoing advancements in ASIC technology and the increasing demand for fast and efficient sequence alignment, ASICs are likely to play a significant role in the future of bioinformatics.

Despite these advances, there are still challenges associated with using hardware for boosting sequence alignment. One major issue is the high data transfer cost among the host computers, which can limit performance. Additionally, oriented to special platform programming requires specialized knowledge, which can be a barrier for many bioinformaticians. However, these hardware accelerators could be rather expensive, lacking the inconvenience of being utilized everywhere. Multiprocessor embedded systems, which are systems where multiple processors are integrated on a single chip, are becoming a prosperous commodity in computing nodes. These systems can provide high performance and energy efficiency, making them well-suited for computationally intensive tasks like sequence alignment.

6 Conclusions and Future Work

To address the memory constraints inherent to embedded processors, we innovatively partitioned the reference genome. This method allowed us to run sequence alignment on embedded processors, each equipped with smaller memory. It is worth noting that this approach does not compromise the accuracy of the results, demonstrating that it is possible to execute such complex computations effectively on systems with limited memory.

The experimental results of our work were promising, showing that our system was able to achieve a 6.8X speedup when utilizing 32 Arm-based processors under a pipeline workflow. This performance enhancement was benchmarked against the original aligner running on an Intel server, emphasizing the significant improvements our embedded solution offers over the traditional processor platform for this specific application.

Despite these encouraging results, we acknowledge that our work is only the initial step in exploring the full potential of Arm-based embedded solutions for genome alignment. In our future research, we aim to extend the applicability of our system to cater to large and long-read genome alignments and to incorporate compatibility with different alignment algorithms. This expansion in capability is anticipated to broaden the scope of our solution, making it applicable to a wider range of genome alignment tasks.

Furthermore, we plan to dig deeper into the genome computational pipeline by exploring the possibility of porting other steps, such as variant calling, to work on embedded processors. By doing so, we aim to create a more comprehensive embedded solution that can handle the entire genome analysis pipeline, thereby increasing the efficiency and speed of genomic data processing.

Acknowledgments. This research is funded by Jiangxi Provincial Natural Science Foundation, with grant number 20212BAB212007.

References

1. Sandes, E.F.O., de Melo, A.C.M.A.: CUDAlign: using GPU to accelerate the comparison of megabase genomic sequences. In: Proceedings of the 15th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 137–146. Association for Computing Machinery, Bangalore, India (2010)
2. Klus, P., et al.: BarraCUDA - a fast short read sequence aligner using graphics processing units. BMC Res Notes **5**, 27 (2012)
3. Liu, Y., Popp, B., Schmidt, B.: CUSHAW3: sensitive and accurate base-space and color-space short-read alignment with hybrid seeding. Plos One **9**(1), e86869 (2014)
4. Accelerating Bioinformatics with NVBIO. https://developer.nvidia.com/blog/accelerating-bioinformatics-nvbio/. Accessed 12 Feb 2015
5. Lindegger, J., Senol Cali, D., Alser, M., Gómez-Luna, J., Ghiasi, N.M., Mutlu, O.: Scrooge: a fast and memory-frugal genomic sequence aligner for CPUs, GPUs, and ASICs. Bioinformatics **39**(5), btad151(2023)
6. Cai, L., Wu, Q., Tang, T., Zhou, Z., Xu, Y.: A design of FPGA acceleration system for myers bit-vector based on OpenCL. In: 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), pp. 305–312. IEEE, Shanghai, China (2019)
7. Chen, Y.L., Chang, B.Y., Yang, C.H., Chiueh, T.D.: A high-throughput FPGA accelerator for short-read mapping of the whole human genome. IEEE Trans. Parallel Distrib. Syst. **32**(6), 1465–1478 (2021)
8. Wirawan, A., Kwoh, C.K., Hieu, N.T., Schmidt, B.: CBESW: sequence alignment on the Playstation 3. BMC Bioinform. **9**(1), 377 (2008)
9. Torres, C., Lloyd, S., Snell, Q.O.: Hardware accelerated sequence alignment with traceback. Int. J. Reconfigurable Comput. **2009**, 76236 (2009)
10. Olson, C.B., Kim, M., Clauson, C., et al.: Hardware acceleration of short read mapping. In: 2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines, pp. 161–168. Toronto, ON, Canada (2012)
11. Stillmaker, A., Bohnenstiehl, B., Stillmaker, L., Baas, B.: Scalable energy-efficient parallel sorting on a fine-grained many-core processor array. J. Parallel Distrib. Comput. **138**, 32–47 (2020)