Future Generation Computer Systems 136 (2022) 221–230

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

nGIA: A novel Greedy Incremental Alignment based algorithm for gene sequence clustering☆

Zhen Ju [a,b], Huiling Zhang [a,b], Jintao Meng [a], Jingjing Zhang [a,b], Jianping Fan [a], Yi Pan [a], Weiguo Liu [c], Xuelei Li [a,*], Yanjie Wei [a,*]

[a] Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, 518005, China
[b] University of Chinese Academy of Sciences, Beijing, 100049, China
[c] Shandong University, Jinan, 250100, China

ARTICLE INFO

Article history:
Received 29 January 2022
Received in revised form 17 April 2022
Accepted 28 May 2022
Available online 6 June 2022

Keywords:
Greedy incremental alignment
CUDA
OneAPI
Gene sequences clustering
Filtering

ABSTRACT

Gene sequence clustering is very basic and important in computational biology and bioinformatics for the study of phylogenetic relationships and gene function prediction, etc. With the rapid growth of the amount of biological data (gene/protein sequences), gene sequence clustering algorithms face more challenges in low precision and efficiency. The growing redundant sequences in gene sequence databases usually contribute to the increasing memory and computing demand for most clustering methods. For example, the original greedy incremental alignment-based (GIA) clustering algorithm obtains high precision clustering results, but with very poor efficiency. Efficient greedy incremental clustering algorithms have been developed with a cost of precision reduction, which usually trade clustering precision off for speed improvement. Algorithms with a better balance between precision and speed are needed. This paper proposes a novel Greedy Incremental Alignment-based algorithm called nGIA for gene clustering with high efficiency and precision. nGIA consists of a pre-filter, a modified short word filter, a new data packing strategy, a modified greedy incremental method, and is parallelized via GPU. The experimental evaluations on four independent datasets show that the proposed tool can cluster datasets with high precisions of 99.99%. Compared with the results of CD-HIT, Vsearch, and Uclust, nGIA is on average 13.6x, 6.2x, and 1.7x faster. In addition, we have developed a multi-node version to handle large data sets. The strong scalability test shows that the multi-node version of nGIA can scale up to 32 threads with a 31% parallel efficiency. The software is available at https://github.com/SIAT-HPCC/gene-sequence-clustering.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

The next-generation sequencing technology has generated data at a higher rate that outpaces Moore's Law, which imposes a big challenge for the research communities that use such genomics resources. For example, gene assembly can be extremely slow due to its huge demand for memory and network communication [1,2], and function predictions for genes can be accelerated by using alignment-free methods are used [3,4]. Therefore, it is of great importance to building non-redundant sequence databases using clustering analysis from massive genomics data. Many previous works about sequence clustering focused on the greedy incremental alignment-based (GIA) algorithm [5–8]. The original GIA algorithm is often time-consuming, thus many better algorithms are proposed. There are two ways to accelerate the GIA algorithm, one is to modify the alignment step, and the other is introducing filters before the alignment.

Other algorithms such as alignment-free algorithms and deep learning-based algorithms can also be used to solve clustering problems [9–11]. Since alignment is the most reliable way to measure sequence similarity, the GIA algorithms are still very popular for sequence clustering.

The efficiency of the GIA algorithms can be improved by adding a pre-filter. The higher the rejection rate of the filter, the faster the GIA-based algorithm can run. However, aggressive

☆ This work was partly supported by the National Key Research and Development Program of China under Grant No. 2018YFB0204403, Strategic Priority CAS Project XDB38050100, the Key Research and Development Project of Guangdong Province under grant No. 2021B0101310002, National Science Foundation of China under grant No. U1813203, the Shenzhen Basic Research Fund under grant No. RCYX20200714114734l9, KQTD20200820113106007 and JSGG20210802154539016, CAS Key Lab, China under grant No. 2011DP173015. We would also like to thank the funding support by the Youth Innovation Promotion Association, China (Y2021101), CAS to Yanjie Wei. We would like to thank Intel for the tech support and resources such as oneAPI DevCloud in this study.

* Corresponding authors.
E-mail addresses: xl.li@siat.ac.cn (X. Li), yj.wei@siat.ac.cn (Y. Wei).

https://doi.org/10.1016/j.future.2022.05.024
0167-739X/© 2022 Elsevier B.V. All rights reserved.

filters generally result in poor clustering precisions. In the worst case, more than 95% of the results are redundant sequences. To design both an efficient and precise GIA clustering algorithm, two factors can be considered: (1) Design a better pre-align filter. Since there is a trade-off between the false-negative rate and the rejection rate of the filter, one needs to carefully design the filter so as not to increase the false-negative rate. (2) Parallel implementation of the algorithm for better efficiency, either via GPU or MPI parallelization, or both.

In this paper, we proposed a precise and efficient GIA clustering tool, called nGIA, by introducing the following modifications to the original algorithm. (1) A pre-filter with a time complexity of O(1) is designed to reduce the number of sequences in the original dataset; (2) A short word-based filter with distance constraint is introduced, to further improve the rejection rates without increasing the false-negative rate; (3) A better data packing strategy is introduced so that more efficient dynamic programming-based alignment can be achieved; (4) The clustering algorithm is implemented via heterogeneous parallelism. The proposed clustering tool is compared with the three most widely-used clustering tools, CD-HIT, Vsearch, and Uclust. The results based on four independent datasets confirm that our tool is the most efficient and also achieves the highest precision. When tested on the four different datasets, the average running times, averaged over all similarity values and all datasets, of CD-HIT, Vsearch, and Uclust are 1264s, 580 s, and 162 s respectively, while the average running time of nGIA is 93 s and is 13.6x, 6.2x and 1.7x faster.

2. Related work

According to whether using an alignment-based algorithm to judge if a pair of sequences are similar, clustering methods can be divided into two categories: alignment-based and alignment-free. Holm proposed the original GIA clustering method in 1998 [5]. Many popular clustering tools/algorithms are based on the similar idea of GIA, such as CD-HIT, Vsearch, and Uclust. Other clustering methods such as machine learning-based or hierarchical clustering are used as well [9–12]. However, the Smith–Waterman-based alignment is still the most popular, ML-based and hierarchical clustering tools often fail to ensure the precision of results.

The original GIA method can generate precise clustering results, but the amount of computation is too demanding [13]. Better algorithms have been proposed to improve the efficiency of GIA-based algorithms. Among all the steps in GIA algorithms, accelerating the alignment step and acceleration of filters before the alignment step are often used to improve the algorithms. The overall workflow of the GIA algorithm and the corresponding acceleration strategies are shown in Fig. 1. Sections 2.1 and 2.2 will describe the related work in detail.

2.1. Accelerations of filtering

A better sequence filter can remove more replicated sequences or sequences with high similarity. By adding a filter before the alignment step of the clustering algorithm, one can reduce greatly the need for the alignment step. In many cases, more than 90% of sequence pairs can be rejected by the filter. Filtering algorithms can be broadly divided into two classes: Locality Sensitive Hashing (LSH) based and Shifted Hamming Distance (SHD) based methods.

At present, some widely used clustering tools use LSH-based filters, such as CD-HIT, Vsearch, and Uclust, which have different clustering mechanisms and are designed to handle different datasets [14]. Li and Godzik implemented CD-HIT, which improved Holm's algorithm by adding a short word filter [13].

The short word filter is inspired by the idea that similar sequences will share some short words. Fu et al. parallelized CD-HIT using multi threads technology [13]. Uclust aligns the top few sequence pairs after a filter step [8], which greatly reduces the amount of calculation in the alignment step and improves the algorithm's efficiency. However, Uclust also increases the false-negative rate using the filter and eventually reduces the precision. Since Uclust is not an open-source software, Rognes et al. developed Vsearch, an open-source alternative to Uclust [7].

There are also many SHD-base filters, but most of them are not popular in the clustering algorithms. Xin et al. proposed an SHD-based filtering algorithm inspired by the pigeonhole principle [15]. Later Chan et al. reduced the computation time of SHD by using Xeon phi for acceleration [16]. Acceleration of SHD algorithm using FPGA is also proposed by Alser et al. [17]. SHD-based filters may have lower time complexity and higher precision. However most SHD-based filters are not as efficient as LSH-based filters.

One can also try to cluster the sequences without alignment by using filters only. Steinager and Söding implemented Linclust, which clusters sequences via finding the same set of k-mers [18]. Without the alignment step, Linclust is the first clustering algorithm which scales linearly.

2.2. Accelerations of alignment

The alignment algorithm is a dynamic programming process with the time complexity of $O(n^2)$ [19]. From our experiments, the alignment can take more than 90% of the running time for sequence clustering algorithms. Therefore, different strategies are used to accelerate the alignment step.

The first category optimizes the alignment algorithm without losing the precision. The Four-Russians algorithm divides the scoring matrix into small blocks and computes each block in advance [20]. This algorithm can theoretically reduce the time complexity of the alignment algorithm to $O(n^2/logn)$. However, the Four-Russians algorithm causes a lot of random memory access and is not suitable for running on GPUs. Loving et al. proposed the BitPal algorithm, which uses one bit to represent a cell in the scoring matrix [21]. The BitPal algorithm can reduce the time complexity of alignment to $O(n^2/m)$, m is the length of the instruction. However, BitPal is limited by the hardware and can only process sequences with a length of less than 512. Based on BitPal, XLCS proposes a more universal algorithm, which can be adapted to any length of sequences and any hardware [22].

The second strategy is using a heuristic algorithm, sacrificing the precision for speed improvement. BLAST first proposes the seed-extend algorithm, which divides the sequence into seeds and aligns subsequences near reference seeds [23]. Wei et al. proposes a method by measuring the sequence similarity by K-tuples [24]. Mash extends the MinHash dimensionality-reduction technique to reduce the computation of clustering [25]. The main idea of all these algorithms is to extract only part of the information in the sequences. Therefore, the less information extracted, the faster the speed, but with a cost of lower precision.

The last strategy relies on heterogeneous acceleration. Since the peak performance of GPUs is more than ten times higher than that of CPUs, it can improve the speed significantly. In 2014 NVIDIA officially proposed NVBIO, a GPU-accelerated library for high-throughput sequence analysis. In addition, many GPU-based alignment algorithms have been developed. For example, Arioc implements the seed-extend algorithm on GPU, and CUDAlign uses multi GPUs to accelerate the alignment algorithm [26,27]. However, these algorithms are not as efficient as NVBIO. On the other hand, GASAL2 outperforms NVBIO by providing specialized, accelerated kernels for local, global, and all types of semi-global alignment [28].

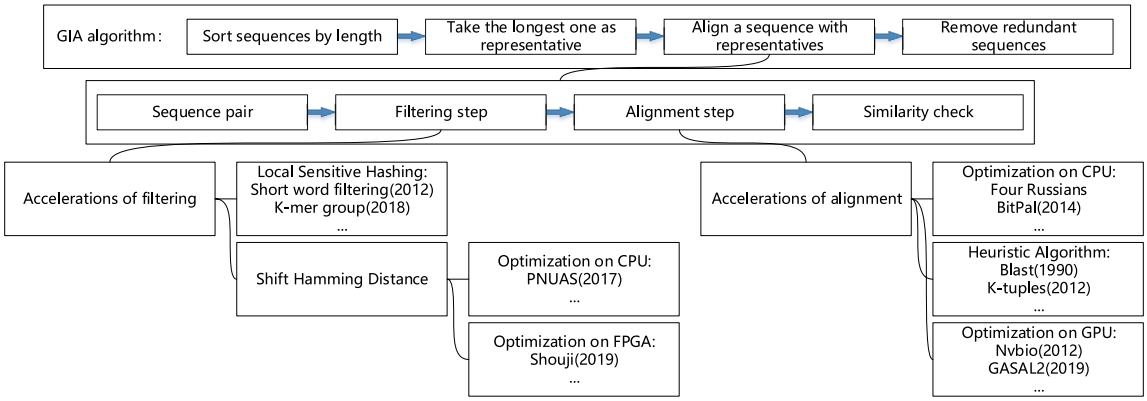


Fig. 1. Overview of GIA algorithm and acceleration strategies.

2.3. Comparison with the original paper

This is an extended paper based on the paper titled "An Efficient Greedy Incremental Sequence Clustering Algorithm", which is accepted by ISBRA2021 conference (17th International Symposium on Bioinformatics Research and Applications, Shenzhen, China, Nov 26–28, 2021) and published by LNCS, volume 13 064 [29]. Compared with the original paper, this extended paper has introduced three additional experiments to better explain and validate our algorithm in this section of "Result and Discussion".

The first experiment compares the amount of computing in terms of sequence alignments and the number of the kernel function calls between GIA and nGIA. The experiment shows that nGIA reduces the average computation of sequence alignments by more than 80% and the number of kernel function calls by more than 90%.

The second experiment calculates the efficiency of the alignment function. The paper compares the efficiency of our alignment function nGIA with the fastest third-party GPU-based sequence alignment method GASAL2, and the results show nGIA is eight times faster than GASAL2.

The third experiment is about the scalability test. Based on the tool implemented in our original paper of ISBRA2021, we have developed a new version that supports multi nodes. The efficiency of the multi node version is more than 30% with 32 nodes.

In addition, we have also extended and modified several sections of the original paper. In the section of "Recent Work", we added the recent development of clustering algorithm from two aspects, accelerations of filtering and alignment. In the section of "Methods and Evaluation Metrics", we added a metric called GCUPS (Giga cell updates per second) for comparing the alignment algorithms and a subsection called "Multi-Node Clustering".

3. Methods and evaluation metrics

The process of the original GIA clustering algorithm is as follows: (1) All sequences are divided into two sets: the representative sequence set and the remaining sequences set. Initially, all sequences are in the remaining sequence set. (2) Sort the remaining sequences by length, and the first one (also the longest one), is taken as the representative sequence. Move the representative sequence to the representative sequence set. (3) Take the next longest sequence in the remaining sequence set and align it with each one in the representative sequence set. If no similar sequences can be found, it is moved to the representative sequence set, otherwise, just remove it. (4) Repeat the above process until the remaining sequence set is empty. The process is shown in Fig. 2.

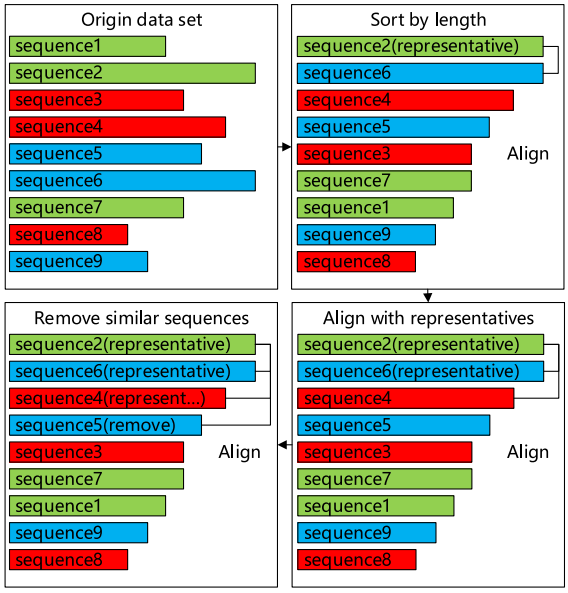


Fig. 2. Process of the original greedy incremental clustering algorithm. A bar represents a sequence. Bars of different lengths represent sequences of different lengths. Sequences of the same color are similar.

The clustering method of nGIA is based on the original GIA algorithm, and four additional modifications are introduced to improve the clustering efficiency. The modifications are base-count pre-filtering, modified short-word-based filtering, data packing, and GPU-based parallelization. In the extended paper, we have also developed a multi-node version of nGIA to handle larger data sets.

3.1. Pre-filtering

The pre-filtering assumes that abundant identical bases exist in similar sequence pairs. For example, sequence "ACCA" and sequence "AAGG" share two same bases (2 As). The detailed algorithm is illustrated in Fig. 3 and Algorithm 1.

Pre-filtering can be divided into two steps. First, count the numbers of As/Cs/Gs/Ts in the text sequence and pattern sequence as "base count". Then adding the minimum of each "base count" in two sequences as the base sum (shown in line 5 of Algorithm . 1). By comparing the base sum with the similarity cutoff, one can determine whether the text sequence and the pattern

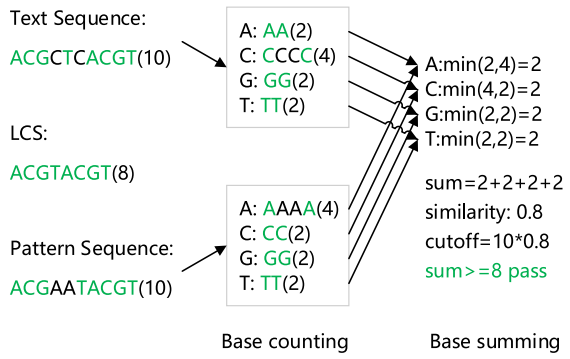


Fig. 3. Illustration of pre-filtering algorithm. Green bases indicate the same bases shared by text and pattern sequences.

sequence are similar or not. In Fig. 3, given a text sequence and a pattern sequence with the length of 10 bases, the length of LCS is 8. The cutoff is equal to the base sum. This pair of sequences pass pre-filtering. The computational complexity of the pre-filtering step is O(1), therefore it can reduce the number of sequences without increasing the computational cost.

Algorithm 1: Pre-filtering

Input:
Base count of sequences: $A[]$, $C[]$, $G[]$, $T[]$
Length of sequences: *Length*[]
Similarity: *Sim*
Output:
Reject this pair of sequences or not: *Reject*

$sumA = min(A[text], A[pattern])$
$sumC = min(C[text], C[pattern])$
$sumG = min(G[text], G[pattern])$
$sumT = min(T[text], T[pattern])$
$baseSum = sumA + sumC + sumG + sumT$
$coutoff = Length[pattern] * Sim$
if $baseSum \leq cutoff$ **then**
  $Reject = False$
else
  $Reject = True$
return *Reject*

3.2. Modified short word filtering

A short word, also called seed, is a subsequence of a fixed number of bases. Many clustering tools rely on short word filtering [6–8], assuming similar gene sequences should share enough short words.

In the original short word filtering, for a short word with length W, a gene sequence with length L contains $(L-W+1)$ short words. Let the similarity be S, then a pair of similar sequences have $L * S$ same bases, and the number of short words that are different between the text and pattern sequences is at most $(L - L * S) * W$. The minimum number of the same short words between the text and pattern sequences can be computed as $(L - W + 1) - (L - L * S) * W$. If the number of the same short words between the two sequences is greater than the minimum, they are considered likely to be similar.

In Fig. 4, according to the traditional short word filtering, the pattern and text sequences are considered likely to be similar since there are 3 same short words which equal to the minimum of 3. However, the length of LCS is 8, which is smaller than the similarity cutoff 9 $(L * S)$, indicating that they are not similar.

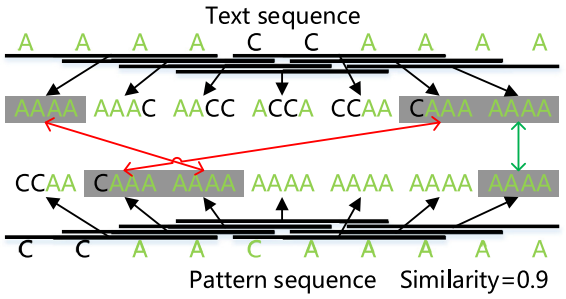


Fig. 4. Modified short word filtering. The length of the text sequence and pattern sequence is 10, and the short word length is 4. Each sequence can generate 7 short words.

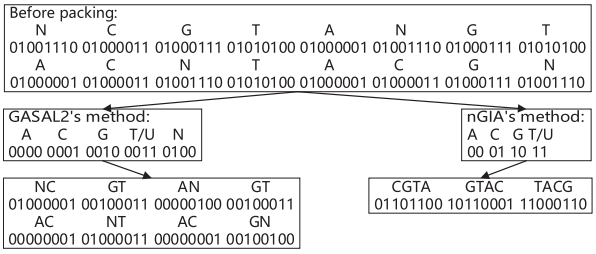


Fig. 5. Illustration of data packing.

To solve the problem, we introduce an additional distance constraint in short word filtering. If the distance between the two same short words on the text and pattern sequences are larger than the allowed distance shift, defined as (1-similarity cutoff)*L, then this pair of the same short words are excluded. For example in Fig. 4, the distance between two CAAAs in text and pattern sequences is 4, larger than the allowed distance shift 1, thus they are excluded. Two out of the three same words in the original short word filtering are excluded after applying the distance constraint (indicated by the red arrows). Since the number of the same words is 1 for the pattern and text sequences, these two sequences are excluded for further analysis.

3.3. Data packing

In sequence clustering, dynamic programming-based sequence alignment consumes over ninety percent of running time. Dynamic programming is memory-intensive and a good data packing technique can significantly improve the speed. DNA and RNA sequences are made up of 5 nucleotides, A, C, G, T/U (T in DNA and U in RNA), and *N* (unknown base), in principle 3 bits are enough for representing the 5 bases [28]. However, in practice, 4 bits representation is widely used, such as in GASAL2, a very fast sequence alignment library [28].

For sequence clustering, the length of LCS is important while nucleotide *N* in the sequence does not affect the result and can be ignored. By ignoring N, one can further pack the data using 2 bits representation, as shown in Fig. 5.

3.4. Heterogeneous parallelization

The original GIA method takes one of the remaining sequences to align with one representative sequence at a time, as shown in Fig. 2. For the property of GPU, we propose a new clustering method that supports parallel computing without changing the clustering results as shown in Fig. 6 and Algorithm 2. The first step of the proposed method is the same as that of the original method and takes the first one as the representative sequence.

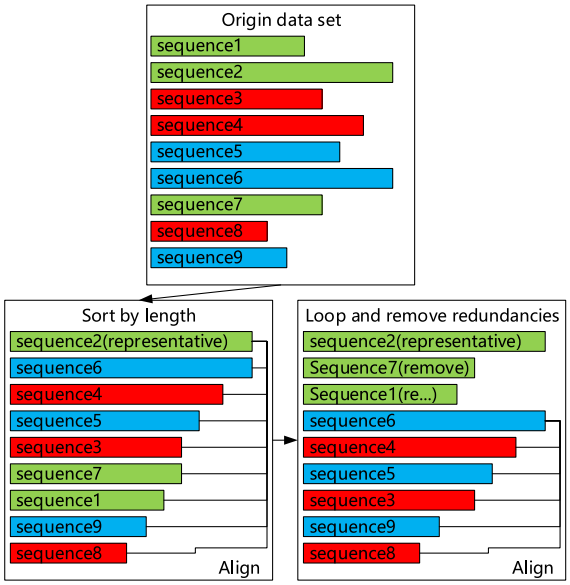


Fig. 6. Illustration of nGIA process. Each bar represents a sequence. Sequences in the same color are similar.

Align all remaining sequences with the representative sequence, and then remove those sequences similar to the representative sequence. Repeat the above process until all redundant sequences are removed.

The proposed method aligns all the remaining sequences with one representative sequence at a time. As shown in Fig. 6 and Algorithm 2, the alignment of representing sequence with other sequences is parallelized, as in the "for loop". Each alignment in the loop is assigned to a thread. Since a fast alignment algorithm on CPU may not be as efficient as for GPU, we have evaluated several alignment algorithms on GPU and found that BitPal [21] and k-band alignment [16] algorithms are efficient on GPU for different datasets.

Algorithm 2: The nGIA process

```
Input:
Sequences: sequences[]
Output:
Cluster result: clusters[]
sort(sequences[])
representative = 0
while sequences ≠ NULL do
    text = sequences[representative]
    # Heterogeneous parallel (one thread on per CUDA
      core)
    for pattern in sequences do
        similar = align(text, pattern)
        if similar then
            remove pattern form sequences
    add text to clusters
    remove text form sequences
return cluster
```

3.5. Multi-node clustering

The nGIA clustering tool relies on a single node that can quickly cluster sequences with high precision, however, clustering of a large dataset is usually limited by the small size of GPU memory. For nGIA, a base requires 4 bytes of GPU memory. A 1 GB data set requires at least 4 GB video memory. To further improve the clustering algorithm, a multi-node version of the clustering algorithm nGIA has been developed to deal with larger datasets. The multi-node nGIA tool uses MPI for communication. First, the sequences are equally divided into different nodes and representative sequences are computed on each node. Then the representative sequences on different nodes are compared for similarity. The result of the overall representative sequences is broadcasted to all the nodes. The software is available at https://github.com/SIAT-HPCC/gene-sequence-clustering.

3.6. Evaluation metrics

The following metrics are used to evaluate the performance of the nGIA method:

(1) Precision for clustering. Precision is the fraction of relevant sequences among the retrieved instances, it can be used to measure the quality of clustering. To generate the gold standard clustering results, sequences are aligned pair by pair with the Smith–Waterman algorithm, and then the redundant sequences in the results can be found. Let the number of redundant sequences in clustering results be R and non-redundant be N. Then we can define precision as below.

$$Precision = \frac{N}{N+R} \tag{1}$$

(2) Rejection rate for pre-filtering. Rejection rate is a measure of filtering efficiency. Let the number of sequence pairs for the pre-filtering step be N_{pre}, and the number of rejected pairs be R_{pre}, then the rejection rate is shown below.

$$RR_{pre} = \frac{R_{pre}}{N_{pre}} \tag{2}$$

(3) Rejection rate improvement for modified short word filtering. It measures the improvement of modified short word filtering compared with the original algorithm. RR_{ori} and RR_{mod} denote the reject rate for the original short word filter and our modified short word filter. The number of sequence pairs that are rejected by the original short word filter and the modified short word filter is denoted as R_s and R_p, respectively. The rejection rate improvement for the modified short word filtering is defined as below.

$$RRI_{mod} = \frac{RR_{mod}}{RR_{ori}} = \frac{R_p}{R_s} \tag{3}$$

(4) Giga cell updates per second (GCUPS). GCUPS is a measure of performance for dynamic programming related alignment algorithm. The most computing intensive part of a dynamic programming algorithm is to calculate a dynamic programming matrix. A cell update is to calculate an element in the matrix, and the number of cell updates and the time (second) are denoted as C and S, respectively. GCUPS can be defined as below.

$$GCUPS = \frac{C}{S} * \frac{1}{1024 * 1024 * 1024} \tag{4}$$

(5) Speed up is used to measure the impact of heterogeneous acceleration. Let the running time of the oneAPI version of our tool on CPU be T_c, and that on GPU be T_g. Speedup is defined as below.

$$Speed\ up = \frac{T_c}{T_g} \tag{5}$$

(6) Strong scaling efficiency is used to measure the efficiency of each node when the number of nodes increases. Let the running time of one node be T_1, and that n nodes be T_n. Efficiency is defined as below.

$$Efficiency = \frac{\frac{T_1}{T_n}}{n} \tag{6}$$

Table 1
Datasets used for evaluation.

Dataset	Count	Length	Size
NCBI	97,413	1251–1598	150 MB
SILVA	381,226	1251–1672	581 MB
GREENGENES	406,997	1251–1829	630 MB
RDP	711,278	1251–1672	1135 MB

Table 2
Hardware environment of CPU, GPU, DCU and oneAPI.

Platform	Processor	Frequency	TDP
CPU	Intel i7-9700K	3.6 GHz	95 W
GPU	NVIDIA GTX1060	1.7 GHz	120 W
OneAPI CPU	Intel i5-1135G7	2.4 GHz	28 W
OneAPI GPU	Intel Iris Xe Max	1.6 GHz	25 W
Multi-node CPU	Hygon C86 7185	2.0 GHz	N/A
Multi-node DCU	Hygon DCU	1.3 GHz	300 W

4. Result and discussion

4.1. Datasets and experimental setting

We have used four widely used 16S rRNA datasets as test data for evaluating the proposed algorithm, as shown in Table 1. The sizes of datasets range from 150 MB to 1.135 GB, with as many as 711,278 sequences. All datasets are downloaded from GREENGENES (https://GREENGENES.lbl.gov).

The nGIA clustering tool is developed in CUDA and the other three tools for comparison are CPU-based. The hardware information of the CPU and GPU platform is shown in Table 2. We selected Intel i7-9700K as the standard, and Nvidia GTX1060, which is similar in price, as the comparison hardware. We also developed an oneAPI version, which allows the same code to run on both the CPU and GPU platforms. For the comparison experiments, CPU and GPU with similar TDP (Thermal Design Power) values were used. The multi-node nGIA tool runs on a cluster built with by Hygon x86 CPUs and DCUs (deep computing units). Detailed hardware information of oneAPI test system and Hygon processors can be found in Table 2.

4.2. Performance improvement by filtering

A pre-filtering step and a modified short word filtering step are applied in the proposed clustering algorithm. Firstly rejection rate RR_{pre} is used for evaluating the pre-filtering, for which the 4 datasets in Table 1 are used as input. As indicated by the solid lines in Fig. 7 when the similarity reaches 0.9, the pre-filter can reject 1% to 5% of the sequences pairs, and when the similarity reaches 0.99, the pre-filter can reject 45% to 53% of the sequence pairs. This indicates that our pre-filter works better for similarity of 0.9 or above. For all 4 datasets, the pattern is similar, the rejection rate of pre-filtering is increasing as the similarity increases. For higher similarity, there are usually less number of sequences that are similar to the representative sequence, thus more sequences are rejected.

The time complexity of the pre-filter is O (1), demonstrated by the fact that in our experiments, the impact of this step on running time is less than 1%. The pre-filtering step is simple yet efficient, and can also be used by any other clustering tools.

The modified short word filter is also compared with the original short word filter. Rejection rate improvement RRI_{mod} is used for comparison, for which the 4 datasets in Table 1 are used as input, and the word length is 4. The results are shown in Fig. 7. As indicated by the dotted lines, for the similarity value smaller than 0.9, the rejection rate improvement RRI_{mod} of the modified filter over the traditional one is less than 1%, and when

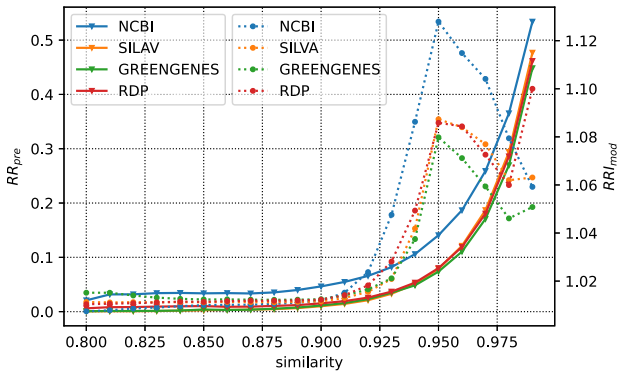


Fig. 7. Left y-axis is rejection rate of pre-filtering on different datasets(solid lines). Right y-axis is rejection rate improvement of the modified short word filter(dotted lines). The x-axis denotes similarity.

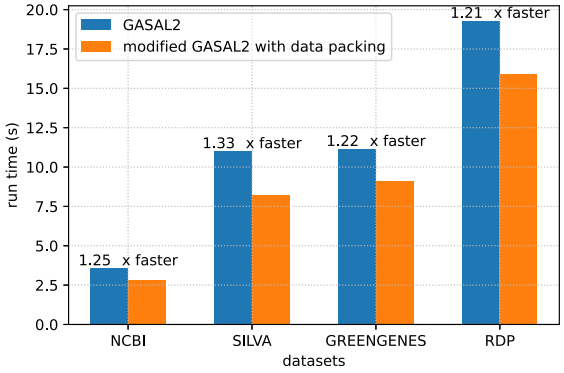


Fig. 8. Running time of the GASAL2 algorithm with modified data packing strategy and the original GASAL2.

the similarity reaches 0.95, RRI_{mod} is 8% to 12%. RRI_{mod} increases with the increasing similarity values when similarity is smaller than 0.95. Since the modified short word filter introduced the allowed distance shift defined by (1-similarity cutoff)*L, which is similarity dependent, a bigger similarity cutoff can result in a very small allowed distance shift, thus making RRI_{mod} decrease when similarity is greater than 0.95.

The time complexity of the improved algorithm is the same as that of the original algorithm. Our improved filter achieves better performance without increasing the cost.

4.3. Performance improvement by data packing

GASAL2 is a very efficient sequence alignment library based on GPU [28]. To evaluate the effect of data packing on the clustering performance, the proposed data packing is implemented in GASAL2, the results of the modified GASAL2 with data packing are compared with the original GASAL2 algorithm. The comparison was performed using NVIDIA GTX1060. Running times of both the original and modified GASAL2 algorithms were compared in Fig. 8. The speedup is defined as the ratio between the running time of GASAL2 and the modified GASAL2 algorithms. As shown in Fig. 8, the speed-ups for NCBI, SILVA, GREENGENES, and RDP datasets are 1.25, 1.33, 1.22, and 1.21 respectively. A minimum of twenty percent speedup can be achieved using the modified data packing strategy in GASAL2. Since packing data does not require redesigning the algorithm, it can be easily applied to other clustering tools.

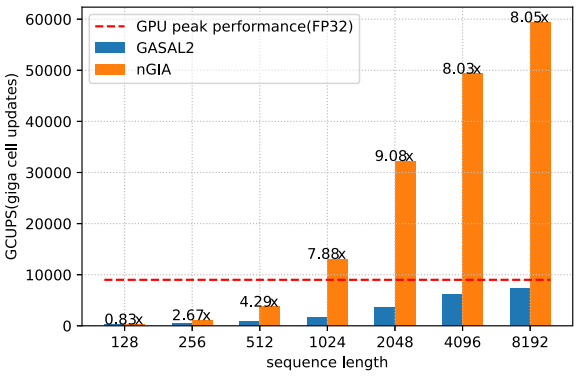


Fig. 9. GCUPS with different lengths and different alignment functions. The dotted line is the peak performance of hardware, in GFLOPS.

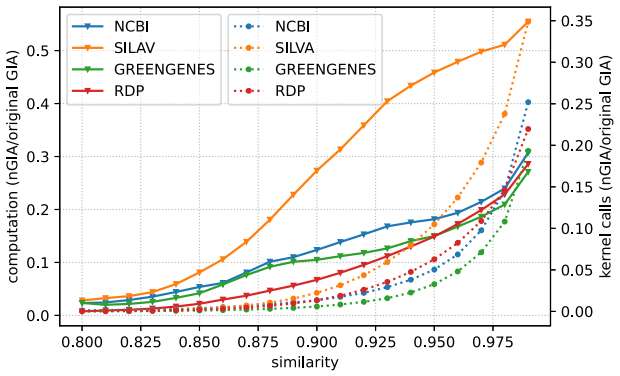


Fig. 10. Left Y-axis is the ratio between the computation of nGIA and that of the original GIA algorithm(solid lines). Right Y-axis is the ratio between the number of the kernel function calls of nGIA and that of the original GIA algorithms (dotted lines).

4.4. Performance improvement of alignment function

From our experience, the alignment step takes more than 90% time on average, so optimizing the alignment step is of great significance. In addition to the use of a better data packaging strategy, we also use the bit-parallel algorithm [21]. Based on the GPU architecture, we implement a fast global alignment function. To evaluate the performance of our alignment function, we compare it with the global alignment function of GASAL2. In the experiment, we generated simulated datasets with different lengths and numbers of sequences. The evaluation focused only on the actual calculation time of the alignment kernel function. The simulated datasets with lengths ranging from 128 to 8192 are tested, and the number of sequences for all datasets is 100,000.

The results are shown in Fig. 9. It can be seen that our global alignment function is about 8x-9x faster than GASAL2 for the datasets when the sequence length is greater than 1000. The performance of alignment increases with the increase of sequence length, which is due to the fact that for longer the sequence length, more data can be reused in the cache, thus higher efficiency can be achieved. Note also that the performance of our method is higher than the peak performance of hardware, as indicated by the dotted red line. The reason is that we use the bit-parallel algorithm, 32 cells can be operated at the same time for each bit operation.

4.5. Heterogeneous parallelism

Section 3.4 introduced how to improve the parallelism of the GIA clustering process. The nGIA aligns each representative sequence with all the remaining sequences, while the original GIA method uses one sequence to compare with all the representative sequences. At the beginning of nGIA, there are fewer representative sequences, the proposed method is more efficient. However, in the later stage of the algorithm, the number of representative sequences is much larger than the remaining sequences and the original algorithm will have better parallelism than nGIA.

To verify and compare the two different approaches, we implemented the above two algorithms. We count the number of the sequence pairs to be aligned as the measurement of computation, and at the same time measure the number of calls of the kernel functions. Using different data sets as input, the results are shown for different similarities in Fig. 10.

It can be seen from Fig. 10 that the nGIA method has not only less computation but also a lower number of kernel function calls, indicated by the smaller ratios for nGIA (less than 1). A smaller ratio value implies a more efficient algorithm. This is because the more sequence pairs in the loop for aligning with the representative sequence, the higher the efficiency. The kernel function needs to be called for each alignment loop. In the original clustering method, the number of the kernel function calls is equal to the number of sequences. While for nGIA, since some sequences will be removed from the remaining sequences in each cycle, the number of kernel function calls will be reduced gradually. Therefore, nGIA achieves higher efficiency. Note that although the amount of computation is reduced greatly compared with the original GIA algorithm, the amount of computation of the nGIA tool is still huge relative to other heuristic-based clustering tools. The heuristic algorithm can reduce the demand for computing by hundreds of times, but it will reduce the precision too.

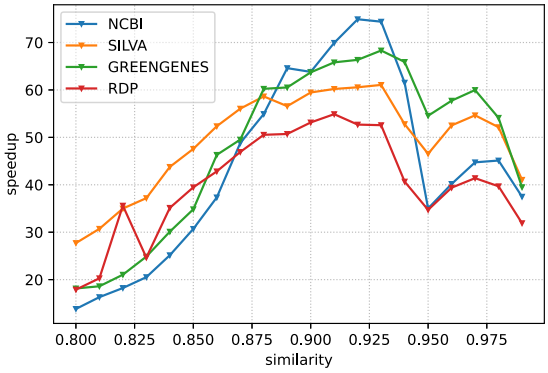


Fig. 11. Heterogeneous Speedup based on different datasets.

To further improve the efficiency of nGIA, heterogeneous acceleration is used in this paper. CUDA and ROCm are the most widely used heterogeneous acceleration platforms, but they support only GPU computing, thus it is difficult to evaluate the impact of migrating serial code to a heterogeneous environment directly. Intel and Khronos SYCL Working Group build a unified open and standard-based programming model, oneAPI, which supports major hardwares and provides a unified programming model and API to users. OneAPI based software/applications can run on CPUs, GPUs, and FPGAs without major code modification. To allow a better comparison, we have implemented an oneAPI version of our clustering algorithm, nGIA. We run oneAPI version of the proposed tool on CPU and GPU (Table 2), and the corresponding speedups are calculated for the four different datasets for comparison. Results are shown in Fig. 11.

It can be seen from Fig. 11 that when tested on the four different datasets, the speedup for GPU vs CPU varies from 14–75

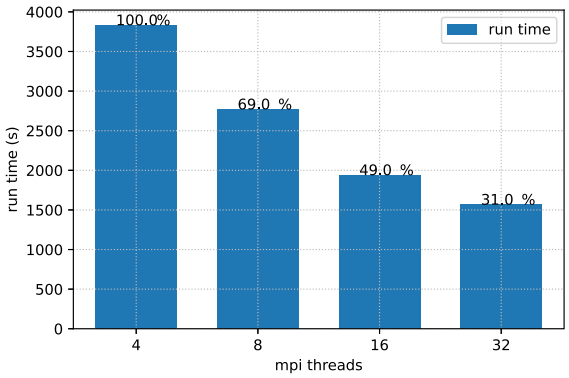


Fig. 12. Scalability of multi-node nGIA tool. The input data set has 5 million sequences, average length of 2000, and the similarity is set to 90%.

times for different similarity values. The maximum speedups of 75, 61, 68, and 55 were achieved for NCBI, SILVA, GREENGENES, and RDP datasets for the similarity values of 0.92, 0.93, 0.93, 0.91, respectively. With the increasing similarity, the computing cost for the filtering step of nGIA is increasing at a faster rate than that for the alignment step. Since the filtering process is I/O intensive and alignment is computing-intensive, the alignment step of nGIA benefits more from the GPU architecture, thus resulting in the overall speedup decrease for larger similarity values.

4.6. Multi-node clustering

The dataset that nGIA can handle is limited by the size of GPU memory. The memory requirement of nGIA is about 4 times the size of the datasets, In this paper, a multi-node version of the nGIA tool has also been developed. In order to test the scalability, a Hygon DCU cluster is used, for which each node is equipped with 4 DCUs. Five million sequences are generated as input data, and the average length of these sequences is 2000.

The result of the scalability test of the multi-node nGIA tool is shown in Fig. 12. The running time for using 4 DCUs is 3831 s, while the running time for using 32 DCUs is 1566 s. Compared with using 4 DCUS, the strong scaling efficiency is about 31% within 32 DCUs.

4.7. Comparison with other clustering tools

We have selected 3 widely used clustering tools, CD-HIT, Vsearch, and Uclust, to compare with nGIA tool. The experiments of CD-HIT, Vsearch, and Uclust were performed using Intel i7-9700K, and the experiment of nGIA was based on NVIDIA GTX1060. The detailed hardware information can be found in Table 2.

Fig. 13 shows the precision (dotted lines) and running time (solid lines) of the compared algorithms on the four different datasets. It can be seen that the precisions of other tools (CD-HIT, Vsearch, and Uclust) improve with the increasing similarity while nGIA stays close to 1 for all similarity thresholds, indicating that regardless of different similarity thresholds, our algorithm can guarantee high precision, measured by comparing the clustering results with the Smith–Waterman algorithm. Compared with CD-HIT, Vsearch and Uclust, it can also be seen that nGIA is the fastest one with smallest running times for all the similarity ranges when tested on the four different datasets.

As shown in Fig. 13, the running time for CD-HIT, Uclust, and nGIA shows peaks around the similarity values of 0.9, 0.87, and 0.96, respectively. This is due to the characteristics of greedy incremental clustering and different filter algorithms used for

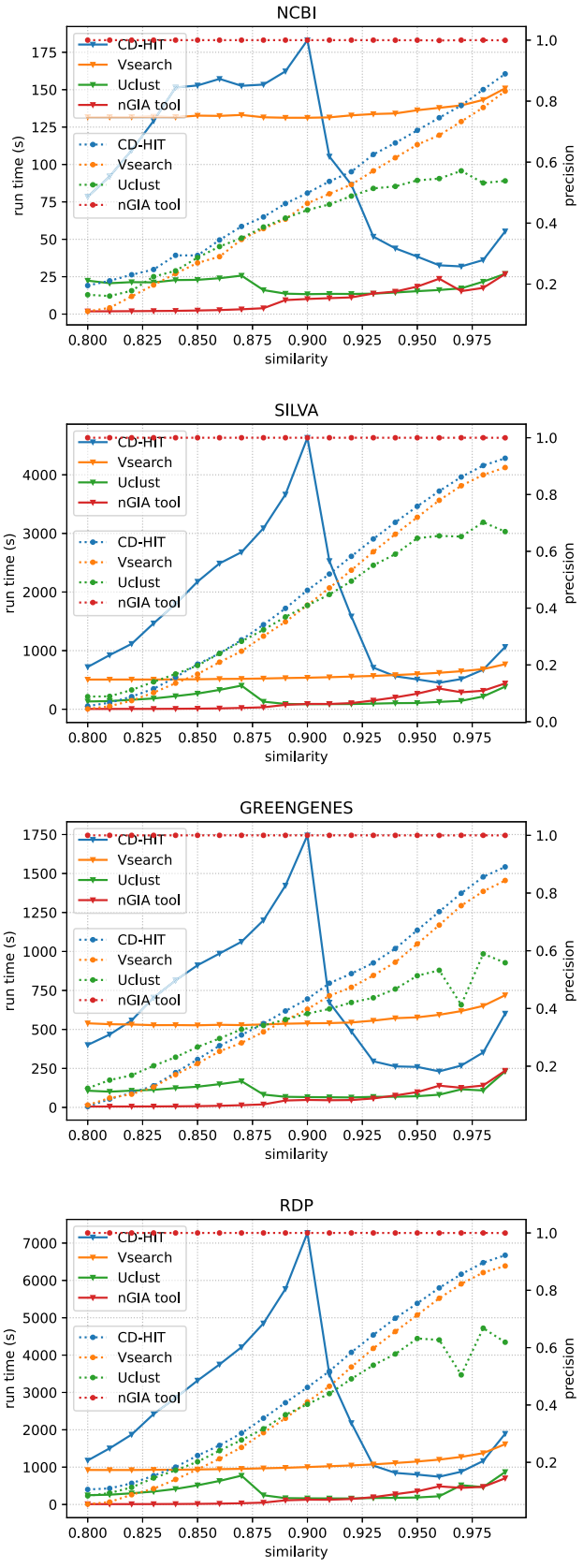


Fig. 13. Running time (solid lines) and precisions (dotted lines) of different tools on different datasets.

clustering. The higher the similarity, the more non-redundant sequences, and the more sequence pairs need to be aligned. But

Table 3
Average running times (seconds) for different clustering methods, last row shows the averages over all datasets.

Dataset	CD-HIT	Vsearch	Uclust	nGIA
NCBI	100	135	19	10
f SILVA	1668	563	175	124
GREENGENES	685	561	105	57
RDP	2602	1065	350	181
Overall	1264	580	162	93

with the increasing similarity, the rejection rate of the filtering algorithm is also increasing, which will reduce the number of sequences pairs to be aligned. For smaller similarity values, the number of sequence pairs to be aligned increases at a faster rate than the number of rejected sequence pairs by the filter algorithm. For bigger similarity values, this trend is reversed. The peak for CD-HIT is more pronounced while for others the peaks are much less pronounced. Overall, CD-HIT took longer time than other three software for all four datasets.

Table 3 shows the average running times for CD-HIT, Vsearch, Uclust, and nGIA when tested on NCBI, SILVA, GREENGENES, and RDP datasets. The average running time of a clustering method running on a specific dataset was calculated by averaging over all the running times for all similarity values. For the largest dataset RDP, the average running times for CD-HIT, Vsearch, Uclust and nGIA are 2602 s, 1065 s, 350 s, and 181 s, respectively, and nGIA is 14.4, 5.9, and 1.9 times faster than CD-HIT, Vsearch, Uclust. While for the smallest dataset NCBI, the average running times for CD-HIT, Vsearch, Uclust and nGIA are 100 s, 135 s, 19 s, and 10 s, respectively, and nGIA is 10, 13.5, 1.9 times faster than CD-HIT, Vsearch, Uclust.

In order to evaluate the overall performance of nGIA compared with other clustering tools, the average running times (last row in Table 3) over all datasets were calculated, and for CD-HIT, Vsearch, and Uclust the overall average running times are 1264 s, 580 s, and 162 s, respectively, while the average running time of nGIA is 93 s and is 13.6x, 6.2x and 1.7x faster than CD-HIT, Vsearch, and Uclust.

5. Conclusion

In this paper, we have proposed an efficient parallel gene sequence clustering algorithm, nGIA. Compared with the original Greedy Incremental Alignment algorithm, nGIA improved the efficiency with high clustering precision by (1) adding a pre-filter with time complexity of O(1), (2) improving the rejection rate of the short word filter; (3) improving the alignment efficiency by a new packing strategy, using two bits to represent a base; (4) modifying the sequence alignment procedure to improve the parallelism and reduce the amount of computation; (5) designing a multi-node version for handling bigger datasets. In addition, we take the advantage of oneAPI programming model and with Data-Parallel C++ to build the code that can run on both CPU and GPU without code reprogramming. The proposed algorithm is superior to three widely used GIA-based tools(CD-HIT, Vsearch, and Vsearch) in both clustering precision and running speed. Compared with the results of CD-HIT, Vsearch and Uclust, nGIA is on average 13.6x, 6.2x and 1.7x faster. The strong scaling parallel efficiency is 31% with 32 threads.

CRediT authorship contribution statement

Zhen Ju: Methodology, Software, Writing – original draft. **Huiling Zhang:** Validation. **Jintao Meng:** Methodology. **Jingjing Zhang:** Investigation, Resources. **Jianping Fan:** Writing – review & editing. **Yi Pan:** Writing – review & editing. **Weiguo Liu:** Writing – review & editing. **Xuelei Li:** project administration. **Yanjie Wei:** conceptualization, Supervision, Writing – review & editing.

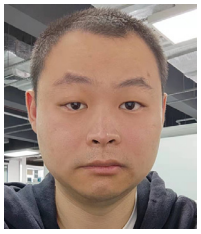
Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yanjie Wei reports financial support was provided by National Key Research and Development Program of China. Yanjie Wei reports financial support was provided by strategic priority CAS Project. Yanjie Wei reports financial support was provided by National Science Foundation of China. Yanjie Wei reports financial support was provided by Shenzhen Basic Research Fund. Yanjie Wei reports financial support was provided by CAS Key Lab.

References

[1] Jintao Meng, Bingqiang Wang, Yanjie Wei, Shengzhong Feng, Pavan Balaji, Swap-assembler: scalable and efficient genome assembly towards thousands of cores, in: BMC Bioinformatics, Vol. 15, BioMed Central, 2014 pp. 1–17.
[2] Jintao Meng, Sangmin Seo, Pavan Balaji, Yanjie Wei, Bingqiang Wang, Shenzhong Feng, Swap-assembler 2: Optimization of de novo genome assembler at extreme scale, in: 2016 45th International Conference on Parallel Processing (ICPP), IEEE, 2016, pp. 195–204.
[3] Dan Wei, Huiling Zhang, Yanjie Wei, Qingshan Jiang, A novel splice site prediction method using support vector machine, J. Comput. Inf. Syst. 9 (20) (2013) 8053–8060.
[4] Dan Wei, Weiwei Zhuang, Qingshan Jiang, Yanjie Wei, A new classification method for human gene splice site prediction, in: International Conference on Health Information Science, Springer, 2012, pp. 121–130.
[5] Liisa Holm, Chris Sander, Removing near-neighbour redundancy from large protein sequence collections., Bioinformatics (Oxford, England) 14 (5) (1998) 423–429.
[6] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, Weizhong Li, CD-HIT: accelerated for clustering the next-generation sequencing data, Bioinformatics 28 (23) (2012) 3150–3152.
[7] Torbjørn Rognes, Tomáš Flouri, Ben Nichols, Christopher Quince, Frédéric Mahé, VSEARCH: a versatile open source tool for metagenomics, PeerJ 4 (2016) e2584.
[8] Robert C. Edgar, Search and clustering orders of magnitude faster than BLAST, Bioinformatics 26 (19) (2010) 2460–2461.
[9] Md Rezaul Karim, Oya Beyan, Achille Zappa, Ivan G Costa, Dietrich Rebholz-Schuhmann, Michael Cochez, Stefan Decker, Deep learning-based clustering approaches for bioinformatics, Brief. Bioinform. 22 (1) (2021) 393–415.
[10] Benjamin T. James, Brian B. Luczak, Hani Z. Girgis, Meshclust: an intelligent tool for clustering DNA sequences, Nucleic Acids Res. 46 (14) (2018) e83.
[11] Yunpeng Cai, Wei Zheng, Jin Yao, Yujie Yang, Volker Mai, Qi Mao, Yijun Sun, ESPRIT-forest: parallel clustering of massive amplicon sequence data in subquadratic time, PLoS Comput. Biol. 13 (4) (2017) e1005518.
[12] Ruilin Li, Xiaoyu He, Chuangchuang Dai, Haidong Zhu, Xianyu Lang, Wei Chen, Xiaodong Li, Dan Zhao, Yu Zhang, Xinyin Han, et al., Gclust: A parallel clustering tool for microbial genomic data, Genom. Proteom. Bioinform. 17 (5) (2019) 496–502.
[13] Weizhong Li, Adam Godzik, Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences, Bioinformatics 22 (13) (2006) 1658–1659.
[14] Quan Zou, Gang Lin, Xingpeng Jiang, Xiangrong Liu, Xiangxiang Zeng, Sequence clustering in bioinformatics: an empirical study, Brief. Bioinform. 21 (1) (2020) 1–10.
[15] Hongyi Xin, John Greth, John Emmons, Gennady Pekhimenko, Carl Kingsford, Can Alkan, Onur Mutlu, Shifted hamming distance: a fast and accurate SIMD-friendly filter to accelerate alignment verification in read mapping, Bioinformatics 31 (10) (2015) 1553–1560.
[16] Yuandong Chan, Kai Xu, Haidong Lan, Bertil Schmidt, Shaoliang Peng, Weiguo Liu, MyPhi: efficient levenshtein distance computation on xeon phi based architectures, Curr. Bioinform. 13 (5) (2018) 479–486.
[17] Mohammed Alser, Hasan Hassan, Akash Kumar, Onur Mutlu, Can Alkan, Shouji: a fast and efficient pre-alignment filter for sequence alignment, Bioinformatics 35 (21) (2019) 4255–4263.
[18] Martin Steinegger, Johannes Söding, Clustering huge protein sequence sets in linear time, Nature Commun. 9 (1) (2018) 1–8.

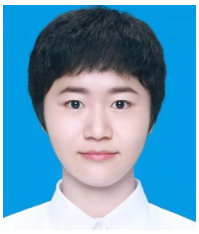
[19] Temple F. Smith, Michael S. Waterman, et al., Identification of common molecular subsequences, J. Mol. Biol. 147 (1) (1981) 195–197.
[20] Vladimir L'vovich Arlazarov, Yefim A Dinitz, MA Kronrod, IgorAleksandrovich Faradzhev, On economical construction of the transitive closure of an oriented graph, in: Doklady Akademii Nauk, Vol. 194, Russian Academy of Sciences, 1970, pp. 487–488.
[21] Joshua Loving, Yozen Hernandez, Gary Benson, BitPAl: a bit-parallel, general integer-scoring sequence alignment algorithm, Bioinformatics 30 (22) (2014) 3166–3173.
[22] Xiaoming Xu, Yuandong Chan, Kai Xu, Jikai Zhang, Xiaoning Wang, Zekun Yin, Weiguo Liu, SLPal: Accelerating long sequence alignment on many-core and multi-core architectures, in: 2020 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), IEEE, 2020, pp. 2242–2249.
[23] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, David J Lipman, Basic local alignment search tool, J. Mol. Biol. 215 (3) (1990) 403–410.
[24] Dan Wei, Qingshan Jiang, Yanjie Wei, Shengrui Wang, A novel hierarchical clustering algorithm for gene sequences, BMC Bioinformatics 13 (1) (2012) 1–15.
[25] Brian D Ondov, Todd J Treangen, Páll Melsted, Adam B Mallonee, Nicholas H Bergman, Sergey Koren, Adam M Phillippy, Mash: fast genome and metagenome distance estimation using MinHash, Genome Biol. 17 (1) (2016) 1–14.
[26] Richard Wilton, Tamas Budavari, Ben Langmead, Sarah J Wheelan, Steven L Salzberg, Alexander S Szalay, Arioc: high-throughput read alignment with GPU-accelerated exploration of the seed-and-extend search space, PeerJ 3 (2015) e808.
[27] Edans Flavius de Oliveira Sandes, Guillermo Miranda, Xavier Martorell, Eduard Ayguade, George Teodoro, Alba Cristina Magalhaes Melo, CUDAlign 4.0: Incremental speculative traceback for exact chromosome-wide alignment in GPU clusters, IEEE Trans. Parallel Distrib. Syst. 27 (10) (2016) 2838–2850.
[28] Nauman Ahmed, Jonathan Lévy, Shanshan Ren, Hamid Mushtaq, Koen Bertels, Zaid Al-Ars, GASAL2: a GPU accelerated sequence alignment library for high-throughput NGS data, BMC Bioinformatics 20 (1) (2019) 1–20.
[29] Zhen Ju, Huiling Zhang, Jingtao Meng, Jingjing Zhang, Xuelei Li, Jianping Fan, Yi Pan, Weiguo Liu, Yanjie Wei, An efficient greedy incremental sequence clustering algorithm, in: International Symposium on Bioinformatics Research and Applications, Springer, 2021, pp. 596–607.



Zhen Ju received the B.S. degree in communication engineering from Lanzhou University of Technology, Lanzhou, China, in 2014, and the M.S. degree in computer applications technology from University of Chinese Academy of Sciences, Beijing, China, in 2016. He is currently pursuing the Ph.D. degree at the University of Chinese Academy of Sciences, Beijing, China. His current research interests include heterogeneous computing and high-performance computing.

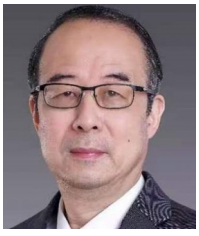
Huiling Zhang received the B.S. degree in communication engineering in 2008 and the M.S. degree in signal and information processing in 2011 from Southwest University, China. She is currently working toward the PhD degree in computer application technology at University of Chinese Academy of Sciences, China. Her research interests include high performance computing, data mining, pattern recognition and bioinformatics.

Jintao Meng received the B.S. and M.S. degrees in computer science from the Central China Normal University, Wuhan, in 2005 and 2008 respectively, and the Ph.D. degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences, Beijing, in 2016. He is an research associate with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He is the author of SWAP-Assembler, and published 20 articles, and 18 inventions. His research interests include high performance computing, bioinformatics, and graph computing.



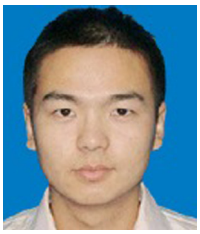
Jingjing Zhang received the B.S. degree in biological science from Northeast Agricultural University, Harbin, China, in 2017, and the M.S. degree in bioinformatics from Shaanxi Normal University, Xi'an, China, in 2020. She is currently pursuing the Ph.D.degree at the University of Chinese Academy of Sciences, Beijing, China. Her current research interest is circular RNAs.

Jianping Fan received his B.S. degree in 1984 from NanKai University in Tianjin, China, and received his PhD degree in Institute of Software, Chinese Academy of Sciences(CAS) in 1990. He worked at institute of computing technology from 1990 to 2006, where he served as the director of National Engineering Center on High Performance Computing and deputy director of institute of computing technology. Since 2006, he serves as Director of Shenzhen Institute of Advanced Technology, CAS. His research interests include high performance computing, Grid computing, and architecture.



Yi Pan (Senior Member, IEEE) received the B.E. and M.E. degrees in computer engineering from Tsinghua University, China, in 1982 and 1984, respectively, and the Ph.D. degree in computer science from the University of Pittsburgh, USA, in 1991. From 2005 to 2020, he served as the Chair of the Computer Science Department, Georgia State University. He has also served as an Interim Associate. He is currently a Chair Professor and the Dean of the College of Computer Science and Control Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China, and a Regents' Professor Emeritus with Georgia State University, USA. His current research interests include bioinformatics and health informatics using big data analytics, cloud computing, and machine learning technologies.

Weiguo Liu received the bachelor's and the master's degrees from Xian JiaoTong University, China, and the PhD degree from Nanyang Technological University, Singapore. He is currently a full professor and the director of High-performance Computing Lab, Shandong University. He was nominated as Taishan Scholar in Shandong, and received numerous awards (e.g., ACM Gordon Bell Prize Award in SC 2017, Fraunhofer IGD Best Paper Award, and CCF HPC China Best Paper Award). He is also on the committee of High Performance Computing, China Computer Federation. His research interests include high-performance computing, bioinformatics, and data mining.



Xuelei Li received the B.S. and M.S. degrees in solid geophysics from Jilin University China, Changchun, China, in 2010 and 2013, respectively, and the Ph.D. degree in solid geophysics from Institute of Geodesy and Geophysics, Chinese Academy of Sciences, Wuhan, China, in 2017. He is currently a research assistant with Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. His research interests include seismic wave propagation and inversion, and high performance computing.

Dr. Yanjie Wei is a professor and the executive director in Center for High Performance Computing, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences. He earned his Ph.D. in 2007 at Michigan Tech University and from 2008 to 2011, he worked as a postdoctoral research associate at Princeton University. His research focuses on high performance computing and computational biology/bioinformatics. He has published more than 100 peer reviewed journal/conference papers.