Bioinformatics, 31(15), 2015, 2475–2481
doi: 10.1093/bioinformatics/btv177
Advance Access Publication Date: 25 March 2015
Original Paper

Sequence analysis

HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy

Quan Zou[1,2,*], Qinghua Hu[1,3], Maozu Guo[3] and Guohua Wang[3,*]

[1]School of Computer Science and Technology, Tianjin University, Tianjin, China, [2]State Key Laboratory of System Bioengineering of the Ministry of Education, Tianjin University, Tianjin, China and [3]School of Computer Science and Technology, Harbin Institute of Technology, Harbin, China

*To whom correspondence should be addressed.

Associate Editor: John Hancock

Received on February 3, 2015; revised on March 10, 2015; accepted on March 23, 2015

Abstract

Motivation: Multiple sequence alignment (MSA) is important work, but bottlenecks arise in the massive MSA of homologous DNA or genome sequences. Most of the available state-of-the-art software tools cannot address large-scale datasets, or they run rather slowly. The similarity of homologous DNA sequences is often ignored. Lack of parallelization is still a challenge for MSA research.

Results: We developed two software tools to address the DNA MSA problem. The first employed trie trees to accelerate the centre star MSA strategy. The expected time complexity was decreased to linear time from square time. To address large-scale data, parallelism was applied using the hadoop platform. Experiments demonstrated the performance of our proposed methods, including their running time, sum-of-pairs scores and scalability. Moreover, we supplied two massive DNA/RNA MSA datasets for further testing and research.

Availability and implementation: The codes, tools and data are accessible free of charge at http://datamining.xmu.edu.cn/software/halign/.

Contact: zouquan@nclab.net or ghwang@hit.edu.cn

1 Introduction

DNA sequence alignment is considered the 'Holy Grail' problem in computational biology and is of vital importance for molecular function prediction. The widely used databases PFAM (Robert *et al.*, 2010) and RFAM (Gardner *et al.*, 2011) are constructed based on multiple sequence alignment (MSA). Molecular function prediction sometimes depends on evolutionary information (Liu *et al.*, 2014). MSA is also required for evolutionary tree reconstruction. Most of the available phylogenetic tree construction software tools require previously aligned sequences as input. When addressing the evolutionary analysis of bacterial and viral genomes, large-scale similar DNA sequences often prevent these MSA tools from functioning (Wang *et al.*, 2013). The evolution of viruses is rapid, and massive viral DNA sequences often appear in phylogenetic reconstructions. Therefore, it is necessary to improve the scalable capacity of MSA tools when analyzing influenza virus DNA (Chang *et al.*, 2007).

The appearance of increasing amounts of DNA and genome data benefits from the improvement of DNA sequencing technology. With the development of the 1000 Genome (Siva, 2008) and HapMap (Manolio and Francis, 2009) Projects, it makes sense to align massive DNA sequences, whose size and length are both scalable.

Most of the recent studies on sequence alignment have focused on mapping and *de novo* assembly. The MSA technique has not undergone any relevant improvements in recent years (Julie *et al.*, 2011; Robert *et al.*, 2006). The available state-of-the-art MSA software tools address DNA and protein sequences equivalently and

ignore the high similarity of DNA sequences. However, we emphasize that the alignment of DNA and protein sequences is not synonymous. The key problem in protein MSA is aligning functional regions. The BLOSUM 62 and PAM250 matrices are referenced for measuring performance. A Hidden Markov Model (HMM) is often employed for detecting shared functional regions (Liu *et al.*, 2009, 2010). In contrast, the core problem in DNA MSA is to match substrings that are as long as possible. Therefore, tree memory structure is always exploited to obtain quick matches (Li *et al.*, 2013; Nilesh and Lucian, 2015).

Several MSA software tools have been developed. In order with data handling size, these tools are T-Coffee (small), CLUSTAL (medium), MUSCLE (medium), MAFFT (medium-large) and Kalign (large), as suggesting by EMBL-EBI (http://www.ebi.ac.uk/Tools/msa/). T-Coffee is a consistency-based MSA tool that attempts to mitigate the pitfalls of progressive alignment methods (Paolo *et al.*, 2011). It only works on protein sequences with a small size. CLUSTAL is a popular MSA tool that employs tree-based progressive alignments (Larkin *et al.*, 2007). It is a multiplatform program that works on both Windows and Linux systems and can address both protein and DNA sequences. The friendly graphic user interface display makes CLUSTAL a popular tool. MUSCLE is well known for its accurate alignment of proteins (Edgar, 2004). MAFFT uses Fast Fourier Transforms, which can run medium-large alignments (Kazutaka and Daron, 2013). To address massive sequences, PASTA divides the set of sequences into several subsets and employs MAFFT for aligning every subset in parallel and has been shown to achieve performance for thousands of RNA sequences (Mirarab *et al.*, 2014). Kalign is a fast MSA tool that concentrates on local regions (Lassmann and Erik, 2005). It is suitable for large alignments. However, our testing suggests that none of these tools can address massive DNA sequences, and they run rather slowly if the count of sequences is greater than 100.

A trie tree is an efficient data structure for storing multiple sequences, and it facilitates the acceleration of searching multiple sequences from a long string (Wang et al., 2010). Therefore, it is helpful for detecting common regions and reducing the time required for alignment. Indeed, various tree-based string storage structures are widely used in DNA searching and mapping, such as BLAT (Kent, 2002) and Hobbes (Ahmadi et al., 2012). However, the storage and acceleration from smart data structure remains a gap in MSA research.

Parallel computing has developed rapidly for scalable data in recent years and has appeared in many bioinformatics applications (Joshua et al., 2014; Mrozek et al., 2014; Zou et al., 2014). However, few parallel software tools have been designed for MSA. Therefore, we developed a parallel MSA software tool based on the hadoop platform for scalable DNA/RNA sequences.

2 Methods

2.1 Centre star strategy

The centre star and progressive tree methods are two basic strategies for MSA. The centre star method runs faster, and it is therefore suitable for the MSA of similar DNA sequences (Zou *et al.*, 2012). The main approach underlying the centre star method is to transform MSA into pairwise alignment based on a 'centre sequence'. This centre sequence is selected, and other sequences are pairwise aligned to the centre sequence. Then, all of the inserted spaces are summed to obtain the final MSA result.

The majority of the running time in pairwise sequence alignment is due to the dynamic programming employed. If the input is similar DNA sequences, long common substrings can be rapidly extracted from the pairwise sequences. Therefore, we only need to align the remaining short regions. The extraction of common substrings can be quickly implemented based on a trie tree data structure, which will greatly reduce the dynamic programming running time for similar DNA sequences.

A trie tree can improve the efficiency of the MSA algorithm. However, it cannot solve the 'big data' problem, due to the fact that the trie trees are always stored in the memory. If the input data increase, the size of trie trees will also markedly increase. Therefore, parallelization is the only fundamental solution for massive data.

The selection of a centre sequence and pairwise alignment consumes most of the running time. We observed that pairwise alignment can be implemented in parallel, and the selection of the centre sequence would cause little difference in performance for the MSA of similar DNA sequences. Therefore, we suggest that parallel hadoop programming can reduce running time and solve the scalable sequences problem for MSA.

2.2 Acceleration with trie trees for similar sequences

The key strategy underlying our method is to detect common substrings before pairwise alignment. The detection process involves linear running time and could decrease the required square running time in dynamic programming. When we find common substrings for pairwise DNA sequences, it only remains to align the remaining regions for pairwise alignment.

The centre star strategy involves three steps: centre star sequence selection, pairwise alignment and subtotalling the inserted spaces.

In the centre star sequence selection step, every sequence is partitioned into several disjoint segments. The segments from every DNA sequence are collected and used to construct a trie tree. The trie tree is similar to a dictionary storing the segments and contains certain indexes. If we want to search for any appearance of any segment in a long sequence, we simply search the long sequence in the trie tree with linear running time, instead of searching each segment individually. Therefore, this process will decrease the amount of time consumed.

When the trie tree is constructed, we search all of the segments in every input DNA sequence. The sequence that contains the most segments will be chosen as the centre star sequence, meaning that it is the sequence that is most similar to all of the others.

After the centre star sequence is chosen, the centre star sequence is pairwise aligned to the other sequences. The matched segments from the trie trees are recorded, and the regions with matched segments need not be aligned. We simply align the remaining regions, as shown in Figure 1. Then, all of the inserted spaces are summed to obtain the final MSA result, which is the same as the original centre star strategy. The flow is shown in Figure 2. There are two points of improvement. First, we partition the DNA sequences and employ a trie tree to accelerate searching, and the centre star sequence can therefore be selected rapidly. Second, matching segments are omitted in the pairwise alignment step, which can reduce the dynamic programming running time.

If there are n DNA sequences with an average length of m, the time complexity for a building trie tree for one sequence is $O(m)$ (Lines 2–3 in Algorithm 1). Searching the n sequences in the trie tree incurs a time cost of $O(nm)$ (Lines 4–6 in Algorithm 1). Therefore, it will require a time of $O(n^2m)$ to find the centre star sequence (Lines 1–9 in Algorithm 1), while the original centre star method will require a time of $O(n^2m^2)$. As $m >> n$, the improved centre star method with trie trees will require less running time.

When the centre star sequence is selected, it still requires pairwise sequence alignment $n-1$ times. The greatest amount of time

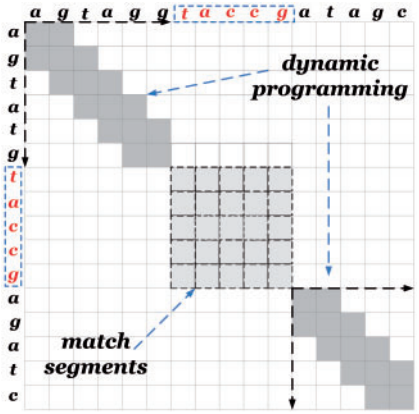

Fig. 1. The dynamic programming in the pairwise alignment step

Algorithm 1. Improved Centre Star Algorithm Based on Trie Trees

Input: n DNA Sequences, S_1, S_2, … S_n
Output: n aligned DNA Sequences S'_1, S'_2, … , S'_n
1. **For** each DNA Sequence, S_i,
2. Partition S_i into k segments $\{S_{i1}, S_{i2}, \ldots S_{ik}\}$ with equal lengths;
3. Construct trie tree $\mathcal{T}_i$ for the segments set $S_i = \{S_{i1}, S_{i2}, \ldots S_{ik}\}$
4. **for** j from 1 to n, j≠i
5. search $\mathcal{T}_i$ in S_j, and set m_{ij} as the segment appearance times; record all of the appearances in $\mathcal{A}_{ij}$
6. end **for**
7. calculate $m_i = \sum_{j=1,j\neq i}^{n} m_{ij}$
8. end **For**
9. $m^* = argmax_{i=1,2,\ldots,n} m_i$, set S_{m^*} as the centre star sequence
10. **For** each i from 1 to n, $i \neq m^*$
11. Partition S_i, and S_{m^*} according $\mathcal{A}_{im^*}$, align the mismatched regions and obtain the pairwise alignment; record all of the positions of inserted spaces in $\mathcal{P}_{im^*}$ and $\mathcal{P}_{m^*i}$.
12. end **For**
13. **For** i from 1 to n, i ≠ m*
14. sum $\mathcal{P}_{m^*i}$ to $\mathcal{P}_{m^*}$
15. end **For**
16. obtain the final result, S'_{m^*}, according to $\mathcal{P}_{m^*}$
17. **For** i from 1 to n, i ≠ m*
18. compare $\mathcal{P}_{m^*i}$ with $\mathcal{P}_m$, and update $\mathcal{P}_{im^*}$, then obtain the final result, S'_i
19. end **For**

consumed is $O(nm^2)$ (Lines 10–12 in Algorithm 1). Because the matching regions need not be aligned, as shown in Figure 1, the expected time complexity is far less than the worst time complexity observed for highly similar DNA sequences.

Summing the inserted spaces will incur a time cost of $O(nm)$ (Lines 13–19 in Algorithm 1). Therefore, the worst time complexity for Algorithm 1 is $O(n^2m) + O(nm^2)$. As $m >> n$, we can consider the worst time complexity to be $O(nm^2)$, which is less than the original centre star algorithm $O(n^2m^2)$. If the input DNA sequences are

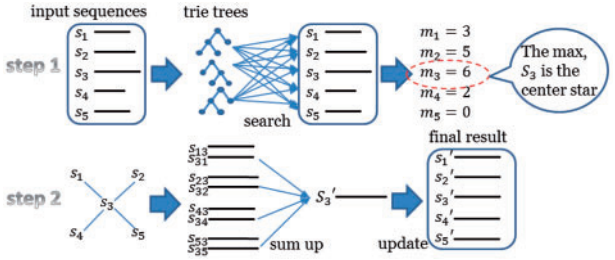

Fig. 2. The flow of the improved MSA algorithm based on trie trees

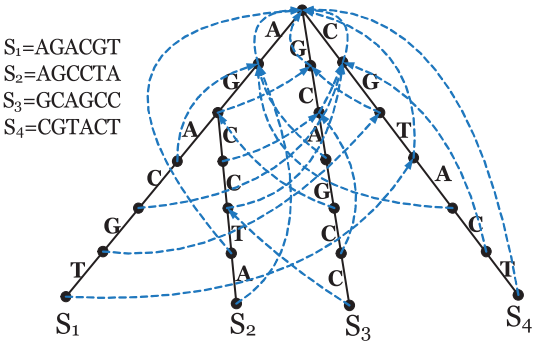

Fig. 3. An example of a trie tree and failure link

highly similar, the expected running time can be $O(n^2m)$. It is known that nm is the total length of the input sequences. Therefore, if $n << m$, the expected time complexity can be viewed as linear.

2.3 Searching a trie tree in linear time

A trie tree is constructed for storing the multiple sequences as an indexed dictionary, which can aid in searching multiple sequences at the same time, rather than individually.

A trie tree contains one root. Every edge is labelled with a nucleotide. Each two edges inherited from the same node cannot be labelled with same nucleotide. Every pattern from the root to the leaf only stands for a sequence, which will be searched. Hence, if we construct a trie tree for k DNA sequences, there will be k leafs.

Failure links can help accelerate the search in linear time. When searching a long sequence in the trie tree, the long sequence will be checked from the root downward. If it reaches a leaf, it means that the stored sequence appears in the long sequence. However, if it is interrupted in the mid nodes, it is undesirable to trace backward for the next search. We only need to check whether a suffix of the interrupted reading frame in the long sequence can match a prefix in the stored sequences.

Thus, we constructed a failure link for every node in the trie tree. For node v in a trie tree, we denote L(v) as the string labelled from the root to v. We denote lp(v) as the longest suffix of L(v); at the same time, lp(v) is a prefix of a stored sequence in this trie tree. Then, the failure link of v will connect to the node representing the prefix lp(v). Figure 3 shows an example of a trie tree and failure link for four DNA sequences.

2.4 Parallelization with hadoop

Although the trie tree based algorithm runs fast, it cannot deal with the big data, which is the key problem for MSA. Here we try to employ MapReduce parallel frame for solving this problem. Entries in Map Reduce are recorded with a (key, value) format. We denoted the key as the sequence name and value as the DNA sequence. Prior

to parallel computing, we pre-process the input sequences and delete illegal characters and unusual sequences. Except for 'AGCTU', the rest characters are viewed as illegal ones. Unusual sequences are a few sequences which are nonhomologous with others. If they are kept, there would be lots of spaces in the alignment result. It would hurt and influence the biology meaning of the alignment. Then, all of the input sequences are formatted as (key, value) pairs for hadoop. Because they are similar sequences, the first sequence is selected as the centre sequence.

In the first stage of the Map function, the data file is automatically divided into several split files, whose size is 64 MB or less. These split files are sent to different Datanodes and aligned to the centre sequence in parallel. After alignment, the centre sequence and the sequence in the split file are updated with inserted spaces. They are still recorded with a (key, value) format, where the key is the sequence name and the value is the two updated aligned sequences.

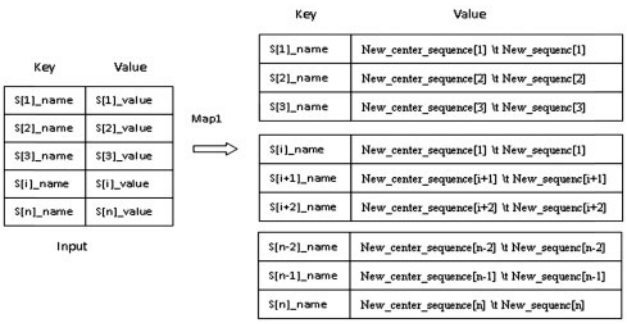

Fig. 4. The input and output of the map function in the first stage

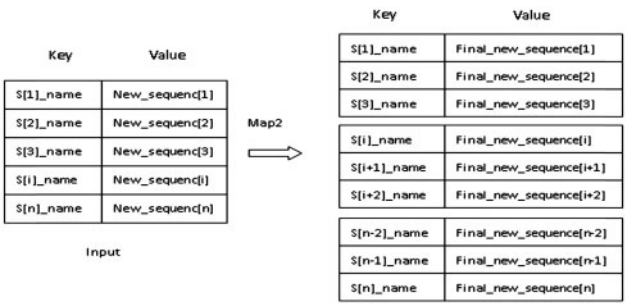

Fig. 5. The input and output of the map function in the second stage

The flow of the map function is shown in Figure 4 and Algorithm 2. Then, the output (key, value) pairs reach the Reduce stage.

In the first stage of the Reduce function, the data are not processed and are output to the HDFS file system directly. Then, the data are collected from the HDFS file system on a local computer, and the aligned centre sequences are extracted and collected. For the n aligned centre sequences, we count the maximum spaces between every two neighbouring characters. The maximum spaces are retained for the Final Centre Sequence.

The second Map-Reduce phase is similar to the first stage. All of the aligned sequences from the first stage are aligned again to the Final Centre Sequence. Because the Final Centre Sequence has the maximum number of spaces between every character, there will be no space inserted into the Final Centre Sequence. Therefore, all of the other sequences will be aligned to the same length as the Final Centre Sequence, which will be the final alignment result. The input and output of the map function in the second stage are shown as Figure 5. Indeed, the original centre star method records the positions of the inserted spaces for the Final Centre Sequence, instead of the second alignment. However, when massive data are dealt with, the distributed storage of the records is a problem. Because they are similar DNA sequences, the k-band alignment is linear time consuming. Therefore, the second Map-Reduce for the alignment is employed. Figure 6 shows the entire flow of the Hadoop MSA strategy.

3 Results

3.1 Data and measurements

Most of MSA studies employ Balibase (Thompson *et al.*, 2005) as the golden benchmark. However, this database is relatively small and is suited only for protein alignment. Because there is no benchmark dataset for addressing the large-scale DNA MSA problem, we employ human mitochondrial genomes (mt genomes) and 16s rRNA. Aligning human mitochondrial genomes is necessary for detecting mtSNP sites, which are associated with Alzheimer's Disease, Parkinson's Disease, and Type 2 Diabetes (Tanaka *et al.*, 2004). The sequence of 16s rRNA is conserved. Therefore, MSAs of 16s rRNA are widely used to infer phylogenetic relationships and to distinguish species in microbial environmental genome analyses (DeSantis *et al.*, 2006; Hao *et al.*, 2011).

In the human mitochondrial genome dataset, there are a total of 672 human mitochondrial genomes, for which the maximum length is 16 579 bp, and the minimum length is 16 556 bp. This is a highly similar dataset. With the aim of testing the performance of our

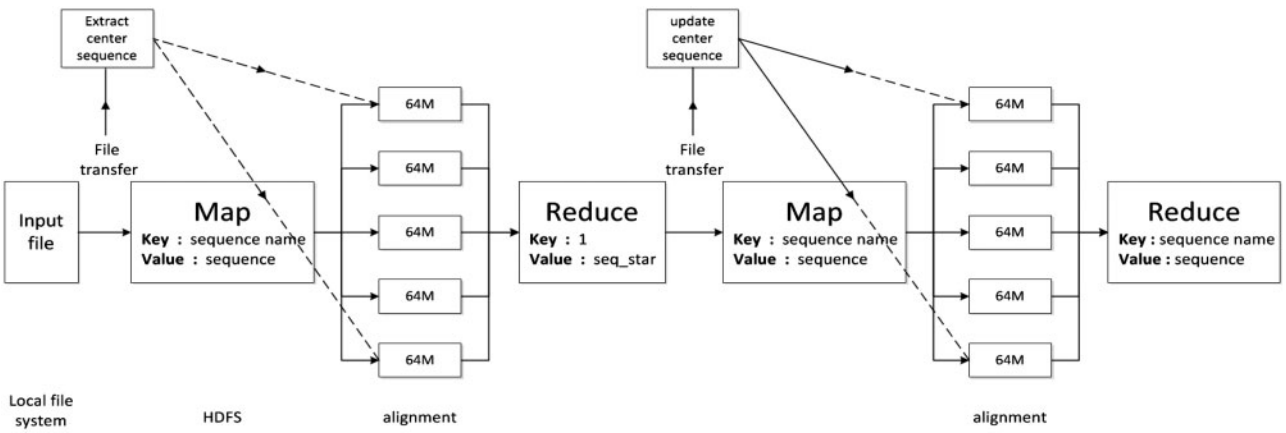

Fig. 6. Flow chart of the Hadoop MSA strategy

Algorithm 2. Function Map_1

For each Map_1(key = sequence_name[i], value = sequence[i])
1: for sequence i ← 1 to Data_Size/64 M **do**
2: key ← sequence_name[i]
3: value ←k-band_alignment_algorithm (center star sequence, sequence[i]);
4:end for

Table 1. Detailed information on the experimental DNA dataset

Dataset	Max length	Min length	Average length	Sequence number	File size
mt genome (1×)	16579	16556	16569.7	672	10 MB
mt genome (20×)				13440	213 MB
mt genome (50×)				33600	532 MB
mt genome (10×x)				67200	1.1 GB
16s rRNA (small)	1599	807	1442.8	108453	156 MB
16s rRNA (big)	1629	807	1388.5	1011621	1.4 GB

Table 2. Time consumption for different software platforms in relation to human mitochondrial genomes with different sizes

	10 M(1×)	213 M(20×)	532 M(50×)	1.1 G(100×)
HAlign (Trie Tree)	3 m 16 s	–	–	–
HAlign (Hadoop)	2 m 21 s	10 m 53 s	14 m 14 s	28 m 28 s
MAFFT	1 m 41 s	175 m	984 m	–
KAlign	170 m 44 s	–	–	–

Table 3. Average SP scores for different software platforms in relation to human mitochondrial genomes with different sizes

	10 M(1×)	213 M(20×)	532 M(50×)	1.1 G(100×)
HAlign (Trie Tree)	183.7	–	–	–
HAlign (Hadoop)	191	191	191	191
MAFFT	152	152	152	–
KAlign	238562	–	–	–

program in relation to large-scale data, we duplicated the mt genomes 20 times, 50 times and 100 times. To address DNA/RNA sequences with low similarity, we also tested our program on two 16s rRNA datasets. The first contained 108 453 16s rRNA sequences with an average length of 1442.8 bp. The longest was 1599 bp, while the shortest was 807 bp. The second dataset included 1 011 621 16s rRNA sequences with an average length of 1388.5 bp. The longest was 1629 bp, and the shortest was 807 bp. The first file was 156 MB, while the second was 1.4 *GB*. Detailed information about the experimental DNA dataset is shown in Table 1.

The main purpose of our work is to accelerate MSA and improve the capacity to handle massive data. Therefore, we focused on the running time for massive data. Moreover, the sum-of-pairs value (Zou *et al.*, 2008) was chosen for measuring the alignment performance. The sum-of-pairs (SP) value is the sum of every pairwise alignment score from the MSA. In the pairwise alignment, if two nucleotides from the same column are different, one is added to, while if a space is inserted, two is added to the score; otherwise if

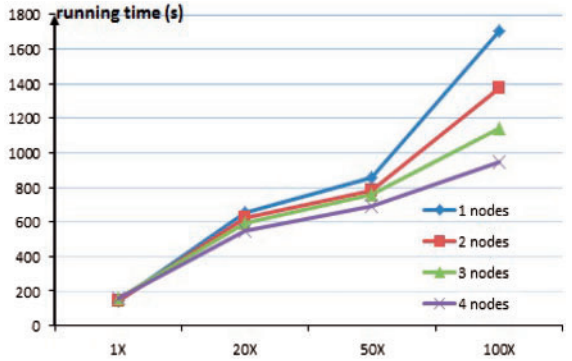

Fig. 7. Running time for mt genome datasets with different Hadoop nodes

Table 4. Time consumption for different software platforms in relation to 16s rRNA sequences with different sizes

	153 M	1.4G
HAlign	54 m 32 s	199 m 35 s
MAFFT	3584 m 52 s	–

Table 5. Average SP scores for different software platforms in relation to 16 s rRNA sequences with different sizes

	153 M	1.4G
HAlign	15660	32079
MAFFT	26743	–
Best Alignment	12889	13100

the two nucleotides are the same, the score is kept constant. Thus, the SP value would be a positive integer, and the lower the SP value, the better the performance. However, the SP value is not suited for massive MSA because the score may be very large and exceed the computer's limitations if the sequence is too large. In our work, we employ the average SP value instead of SP. The average SP is the SP divided by the number of sequences, *n*. The average SP can also describe alignment performance.

3.2 Comparison with state-of-the-art tools

Most of the available state-of-the-art MSA software tools cannot address large-scale data. Therefore, we only performed comparisons with MAFFT, KAlign and PASTA. The experiments were implemented based on clusters, in which the node had 64 GB of memory, a 3.6 GHz 4 core CPU and a 64 bit Ubuntu Operation System. KAlign and MAFFT were run on single node without any parallel operation for fairness, and the Hadoop version of HAlign was also tested on only one node.

Table 2 shows the time consumed for the human mitochondrial genome dataset. From Table 2, we can see that MAFFT, PASTA and KAlign cannot address large files of greater than 1 GB. However, HAlign (Hadoop version) requires less than half an hour. Furthermore, as the nodes grow, it will save more running time, as shown in Figure 7 and analyzed in 3.3.

Table 3 shows the comparison of the average SP values. Because the human mitochondrial genome dataset is highly similar, the performance of the different types of software makes little difference. Only KAlign produced poor results. It was found that KAlign inserted many spaces before the aligned sequences, which might be a bug within KAlign. We can also conclude that the Hadoop version

of HAlign works well on the duplicated files. The average SP score did not increase for the duplication, which demonstrated the robustness of our software.

Because the human mitochondrial genomes are highly similar, HAlign saves time in k-band dynamic programming. We also tested performance for datasets with low similarity. The 16s rRNA datasets are massive less similar. Original work (DeSantis *et al.*, 2006) demonstrated the best alignment with artificial assistance. We deleted all of the spaces from the original data and performed MSA again using HAlign (Hadoop version) and MAFFT.

KAlign cannot be used for files larger than 100 MB. Thus, we compared the observed running time with MAFFT. MAFFT also cannot be used for large files greater than 1 GB, and it ran for nearly 60 h for the 153 MB file. In contrast, HAlign ran for less than 1 hour, as shown in Table 4. The alignment performance based on SP scores is shown in Table 5.

From Table 5, we can observe that HAlign produced better alignment results than MAFFT. The average SP score for the HAlign results was far lower than that obtained using MAFFT (15660 vs. 26793). HAlign compares favourably with the best alignment results (15660 vs. 12889). The best alignment is downloaded from the original web site, which is done with manual work. Even for the very large dataset (∼1.4 GB) with less similarity, HAlign still functioned moderately. Therefore, we confirm the robustness of HAlign, whether using scalable data or a tiny dataset, with high- or low-similarity sequences.

The Hadoop version of HAlign, on only one node, can outperform the trie tree version regarding the running time for small datasets because the Hadoop platform can employ all cores in a CPU. However, the trie tree version is only a single-threaded tool. Nevertheless, the results of the trie tree version are not meaningless, as this version works well for small datasets. More generally, it does not require the Hadoop platform, and it is therefore much more user-friendly for junior users.

3.3 Speedup of hadoop software

To test the scalability and speedup of the Hadoop version of HAlign, we tested the 1×, 20×, 50× and 100× human mt genome datasets with 1–4 nodes in clusters. When the dataset was fixed and the cluster size was varied, it can be seen from Figure 7 that the running time showed little difference for the 1× dataset (∼10 MB), whereas it differed markedly for the 100× dataset (>1 GB) because that data transfer among the clusters also consumes time. Therefore, if the dataset is small and the running time is only a few minutes, the Hadoop cluster cannot show its speedup ability. When the dataset is large and scalable and the running time is several minutes or longer, the parallel mechanism exhibits its force. The inclusion of more nodes in the cluster can save the running time through parallel scheduling of Hadoop.

Moreover, we can observe the scalable data processing ability by fixing the cluster nodes and varying the size of the dataset. Other software tools cannot address a dataset of more than 1 GB, while the Hadoop version of HAlign not only calculated the 100× dataset, but also required less than twice the time compared with the 50× dataset.

4 Conclusion

MSA is an important and fundamental tool in bioinformatics, especially for phylogenetic tree reconstruction. There are complex interrelationships between MSA and phylogenetic tree reconstruction, involving mutual promotion and restraint. Phylogenetic tree reconstruction algorithms always require MSA results as input data, while MSA algorithms sometimes require phylogenetic trees as guidelines. For massive unaligned DNA sequences, several MSA free phylogenetic tree reconstruction algorithms have been proposed (Cheng *et al.*, 2013; Leimeister *et al.*, 2014; Sims *et al.*, 2009). The core strategy of these algorithms is to calculate string distance based on the longest common substrings or word frequency, instead of sequence alignment. This strategy is similar to our trie tree-based algorithm. This method would be better suited for DNA sequences than protein sequences.

In this work, we developed two MSA tools based on the centre star strategy. The first tool employed trie trees to accelerate the MSA of highly similar DNA sequences. It runs fast but cannot deal with the big data. The second was designed to deal with scalable data. It works in parallel with Hadoop, which is an open platform for parallel programming that has developed rapidly in recent years. Experiments using files >1 GB demonstrated the ability of HAlign.

From the experiments, we can conclude that HAlign outperforms the available state-of-the-art MSA software tools regarding its ability to handle large-scale DNA/RNA datasets, and it is suited for massive highly similar DNA sequences. It also runs faster for sequences of low similarity because it is a software tool that runs in parallel. Here, we simply employed dynamic programming for basic alignment. RNA structure information and a protein substitution matrix were not considered because they are time consuming. Taking such data into consideration will be future work to be performed for HAlign.

Both the single thread and the parallel tools are coded with Java, which works on multiple operation systems. Hadoop 2.0 is required for the parallel tool.

We have constructed a web site http://datamining.xmu.edu.cn/software/halign/ for sharing the software tools and codes.

Funding

This work was supported by the Natural Science Foundation of China (61370010, 61371179, 61271346, 61222210, 61432011), the New Century Excellent Talents Support Program from the Ministry of Education, China (NCET-13-0176).

Conflict of Interest: none declared.

References

Ahmadi,A. *et al.* (2012) Hobbes: optimized gram-based methods for efficient read alignment. *Nucleic Acids Res.*, **40**, e41.

Chang,S. *et al.* (2007) Influenza Virus Database (IVDB): an integrated information resource and analysis platform for influenza virus research. *Nucleic Acids Res.*, **35**, D376–D380.

Cheng,J. *et al.* (2013) AGP: a multi-methods web server for alignment-free genome phylogeny. *Mol. Biol. Evol.*, **30**, 1032–1037.

DeSantis,T.Z. *et al.* (2006) NAST: a multiple sequence alignment server for comparative analysis of 16S rRNA genes. *Nucleic Acids Res.*, **34**, W394–W399.

Edgar,R.C. (2004) MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Res.*, **32**, 1792–1797.

Gardner,P.P. *et al.* (2011) Rfam: Wikipedia, clans and the "decimal" release. *Nucleic Acids Res.*, **39**, D141–D145.

Hao,X. *et al.* (2011) Clustering 16S rRNA for OTU prediction: a method of unsupervised Bayesian clustering. *Bioinformatics*, **27**, 611–618.

Joshua,L. *et al.* (2014) BitPAI: a bit-parallel, general integer-scoring sequence alignment algorithm. *Bioinformatics*, **30**, 3166–3173.

Julie,D.T. *et al.* (2011) A comprehensive benchmark study of multiple sequence alignment methods: current challenges and future perspectives. *PLoS ONE*, **6**, e18093.

Kazutaka,K. and Daron,M.S. (2013) MAFFT Multiple Sequence Alignment Software Version 7: improvements in performance and usability. *Mol. Biol. Evol.*, **30**, 772–780.

Kent,W.J. (2002) BLAT—the BLAST-like alignment tool. *Genome Res.*, **12**, 656–664.

Larkin,M. A. *et al*. (2007) Clustal W and Clustal X version 2.0. *Bioinformatics*, **23**, 2947–2948.

Lassmann,T. and Erik,L.S. (2005) Kalign—an accurate and fast multiple sequence alignment algorithm. *BMC Bioinformatics*, **6**, 298.

Leimeister,C.A. *et al*. (2014) Fast alignment-free sequence comparison using spaced-word frequencies. *Bioinformatics*, **30**, 1991–1999.

Li,G. *et al*. (2013) A partition-based method for string similarity joins with edit-distance constraints. *ACM Trans. Database Syst.*, **38**, 9.

Liu,B. *et al.* (2009) Prediction of protein binding sites in protein structures using hidden Markov support vector machine. *BMC Bioinformatics*, **10**, 381.

Liu,Y. *et al.* (2010) MSAProbs: multiple sequence alignment based on pair hidden Markov models and partition function posterior probabilities. *Bioinformatics*. **26**, 1958–1964.

Liu,B. *et al*. (2014) Combining evolutionary information extracted from frequency profiles with sequence-based kernels for protein remote homology detection. *Bioinformatics*, **30**, 472–479.

Manolio,T.A. and Francis,S.C. (2009) The HapMap and genome-wide association studies in diagnosis and therapy. *Annu. Rev. Med.*, **60**, 443.

Mirarab,S. *et al*. (2014) PASTA: ultra-large multiple sequence alignment. In: Sharan,R. (Ed.), *Research in Computational Molecular Biology (RECOMB)*. Springer International Publishing Switzerland, pp. 177–191.

Mrozek,D. *et al*. (2014) Cloud4Psi: cloud computing for 3D protein structure similarity searching. *Bioinformatics*, **30**, 2822–2825.

Nilesh,K. and Lucian,I. (2015) E-MEM: efficient computational of maximal exact matches for very large genomes. *Bioinformatics*, **31**, 509–514.

Paolo,D.T. *et al*. (2011) T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension. *Nucleic Acids Res.*, **39**, W13–W17.

Robert,C.E. and Serafim,B. (2006) Multiple sequence alignment. *Curr. Opin. Struct. Biol.*, **16**, 368–373.

Robert,D.F. *et al*. (2010) The Pfam protein families database. Nucleic Acids Res., **38**, D211–D222.

Sims,G.E. *et al*. (2009) Alignment-free genome comparison with feature frequency profiles (FFP) and optimal resolutions. *Proc. Natl Acad. Sci.*, **106**, 2677–2682.

Siva,N. (2008) 1000 Genomes project. *Nat. Biotechnol.*, **26**, 256.

Tanaka,M. *et al*. (2004) Mitochondrial genome variation in eastern Asia and the peopling of Japan. *Genome Res.*, **14**, 1832–1850.

Thompson,J.D. *et al.* (2005) BAliBASE 3.0: latest developments of the multiple sequence alignment benchmark. *Proteins: Struct. Funct. Bioinf.*, **61**, 127–136.

Wang,J. *et al*. (2010) Interactive and fuzzy search: a dynamic way to explore MEDLINE. *Bioinformatics*, **26**, 2313–2320.

Wang,J. *et al*. (2013) Lnetwork: an efficient and effective method for constructing phylogenetic networks. *Bioinformatics*, **29**, 2269–2276.

Zou,Q. *et al*. (2008) An algorithm for DNA multiple sequence alignment based on center star method and keyword tree. *Acta Electronica Sinica*. **37**, 1746–1750.

Zou,Q. *et al*. (2012) A novel center star multiple sequence alignment algorithm based on affine gap penalty and K-band. *Physics Procedia*. **33**, 322–327.

Zou,Q. *et al*. (2014) Survey of MapReduce frame operation in bioinformatics. *Brief. Bioinf.*, **15**, 637–647.