Journal of Computational Biology: http://mc.manuscriptcentral.com/liebert/jcb

MASC: A Linear Method for Multiple Nucleotide Sequence Alignment on Spark Parallel Framework

Journal:	*Journal of Computational Biology*
Manuscript ID	JCB-2017-0040
Manuscript Type:	Method Article
Keyword:	MULTIPLE ALIGNMENT
Manuscript Keywords (Search Terms):	Multiple Sequence Alignment, Suffix Tree, Center-Star Strategy, Spark
Abstract:	Multiple sequence alignment (MSA) is an essential prerequisite and dominant method to deduce the biological facts from a set of molecular biological sequences. In this work, we take advantage of a center-star strategy to reduce the MSA problem to pairwise alignments, and we use a suffix tree to match identical substrings between the two pairwise sequences. We can accomplish MSA in O(mn) time in this manner, where m is the number of sequences, and n is the average length of the sequences. Furthermore, we execute our method on the Spark distributed parallel framework to deal with ever-increasing massive datasets. Our method is significantly faster than previous techniques, with no loss in accuracy for highly similar nucleotide sequences like homologous sequences, which we experimentally demonstrate. This work will facilitate 16S rRNA metagenomics analyses and other research.

MASC:

A Linear Method for Multiple Nucleotide Sequence Alignment on Spark Parallel Framework

Wenhe SU, School of Computer Science and Technology, National University of Defense Technology, Changsha, China

Quan ZOU, School of Computer Science and Technology, Tianjin University, Tianjin, China

Xiangke LIAO, School of Computer Science and Technology, National University of Defense Technology, Changsha, China

Yutong LU, School of Computer Science and Technology, National University of Defense Technology, Changsha, China

Shaoliang PENG*, School of Computer Science and Technology, National University of Defense Technology, Changsha, China

*Corresponding author: E-mail: pengshaoliang@nudt.edu.cn

Connect Imformation:

Name	Telephone	E-mail
Wenhe SU	+86 13272094630	suwenhecn@gmail.com
Quan ZOU	+86 17092261008	zouquan@nclab.net
Xiangke LIAO	+0731 84576322	xkliao@nudt.edu.cn
Yutong LU	+0731 84576322	ytlu@nudt.edu.cn
Shaoliang PENG	+86 13574817196	pengshaoliang@nudt.edu.cn

Abstract

Multiple sequence alignment (MSA) is an essential prerequisite and dominant method to deduce the biological facts from a set of molecular biological sequences. In this work, we take advantage of a center-star strategy to reduce the MSA problem to pairwise alignments, and we use a suffix tree to match identical substrings between the two pairwise sequences. We can accomplish MSA in O(*mn*) time in this manner, where *m* is the number of sequences, and *n* is the average length of the sequences. Furthermore, we execute our method on the Spark distributed parallel framework to deal with ever-increasing massive datasets. Our method is significantly faster than previous techniques, with no loss in accuracy for highly similar nucleotide sequences like homologous sequences, which we experimentally demonstrate. This work will facilitate 16S rRNA metagenomics analyses and other research.

Key words: Multiple Sequence Alignment, Suffix Tree, Center-Star Strategy, Spark

太笼统，还是不知道MSA是什么

center-star 的优势是什么不知道，连她是啥我都不知道，可以加一个对它进行表述的定语或同位语

变换成了成对的校准问题有什么好处，和之前的最大的区别是什么不知道；这里给我的感觉是这个方法优势在于把原问题归约了成对校准，但是读后文发现所有启发算法都是要这样他，so，怪怪的

不知道match这些是什么意义，用来干嘛的；发现后文也并没有具体解释如何使用树进行match的

加个线性的这种关键词我觉得更好，总结了数学特点

1. Introduction

Multiple sequence alignment (MSA) is an essential preprocess to many bioinformatics analyses, including homology modeling, secondary structure prediction, phylogenetic reconstruction, and protein structure and function prediction. MSA is the alignment of more than two molecular biological sequences, aiming to discover maximally similar (or identical) amino acid or identical nitrogenous base positions across the aligned query set of sequences. An MSA is visualized as a two-dimensional matrix in which the rows are the individual sequences and the columns are maximally similar or identical amino acid or nitrogenous base positions arranged to correspond by inserting gap characters in appropriate positions (known as indels), such that the biological relationship of the sequences is best characterized. An MSA can provide a wealth of information about the structure/function relationships within a set of sequences, such as the evolutionary conservation of functionally or structurally important sites, and conserved hydrophobicity patterns in precise regions.

Although dynamic programming using the Needleman–Wunsch algorithm (Needleman and Wunsch, 1970) can be generalized in theory to produce an alignment for any number of sequences, unfortunately, this generalization leads to an explosive increase in computer time and memory requirements as the number of sequences increases(Taylor, 1990). MSA remains under continuous development, and is regarded as one of the most challenging problems in the field of bioinformatics and computational biology. Furthermore, optimal MSA solution is NP-complete

校准是什么还是不明白

什么是对齐的序列的查询集合

上面说MSA是校准，下面有说又说是一个表达各个序列上什么相似或者相同什么鬼的位置的矩阵

无法理解；读了全文还是不理解这个矩阵是什么鬼。

一个序列的集合?

在一个集合之内的关系?

例子也没有明白是啥。。

(Wang and Jiang, 1994) which means the problem cannot be solved in polynomial time. Therefore, most of the algorithms currently utilized are heuristic and combinatorial optimization algorithms for obtaining the approximation of optimal solution.

We developed a new method that performs pairwise alignment in O(*n*) time between two highly similar sequences based on a suffix tree structure, which is well-known for its string processing power. Here, we take two sequences which have at least 56.8 percent nucleotides in common as highly similar sequences. We also designed and implemented a novel MSA method called MASC (Multiple sequence Alignment based on a Suffix tree and Center-star strategy), which has extremely high performance and accuracy with our new pairwise alignment algorithm. Furthermore, we implemented MASC on the Spark parallel distributed framework for use with massive scale sequence data and accelerate the method by using multiple computing node parallel programming.

摘要里是先MSA转换为成对校准，然后（你用的and）利用后缀树讲相同子序列的match起来
但是这里的逻辑是基于后缀树的结构来表现为成对的校准，方法的前后顺序搞不懂来。。

是also吗，你优化了两个算法吗，那MASC方法的主要内容是什么？

2. Related Work

The most commonly used heuristic methods involve a progressive-alignment strategy, iterative-alignment strategy, or center-star strategy. ClustalW (Thompson *et al.*, 1994) is the most widely used implementation of the progressive-alignment strategy. MAFFT (Lounkine *et al.*, 2012) is quite fast, using a fast Fourier transform algorithm along with a progressive-alignment strategy.

Progressive-alignment generates a quasi-phylogenetic guide tree among the sequences and gradually builds up the alignment in a pairwise fashion, following the order provided by the tree. Although successful for a wide variety of cases, the method suffers from greediness. Errors made in initial alignments cannot be rectified later as the remainder of the sequences are added (Notredame, C., Higgins, D. G., & Heringa *et al.*, 2000). Progressive alignment is also quite time-consuming.

The iterative strategy is an interesting alternative method. Iterative strategies do not provide any guarantee of an optimal solution, but are reasonably robust, and are much less sensitive to the number of sequences than their deterministic counterparts (Gotoh, 1996). PRRN (Gotoh, 1996) and MUSCLE (Edgar, 2004) employ an iterative-alignment strategy.

既然用到了，是不是应该说明一下这到底是什么策略，否则我这种水平的人体会不到它为什么可以带来这些性能

The star alignment strategy, which is utilized in our project, is a fast method for solving MSA, and suitable for highly similar sequences. The primary time cost in the star alignment strategy occurs during the construction of the similarity matrix, and in

discovering the center sequence. Assume that it takes O(t) to run a pairwise alignment between two sequences. The complexity of the algorithm is O($m^2t + mt$). Therefore, running MSA on a set of homologous sequences using the star alignment strategy reduces the time cost to O(mt), which is much faster than both the progressive and iterative strategies.

t是？m是？
为什么最后得到了
0（mt）？

Regardless of which heuristic method used, the main simplification idea common to all is to reduce the MSA to a series of pairwise sequence alignments. Consequently, pairwise alignment is a dominant component of all MSA techniques. Traditional dynamic programming algorithms, such as Smith–Waterman (Smith and Waterman, 1981) and Needleman–Wunsch (Needleman and Wunsch, 1970), require O(n^2) temporal and spatial complexity to perform pairwise alignment, where n is the maximum length of two sequences. Other, faster algorithms have been developed, such as MAFFT (Lounkine *et al.*, 2012), which is O($n\log n$). These algorithms work extremely well on conventional tasks with multiple single protein sequences, cDNA sequences, or relatively short genomic DNA sequences containing a single gene and simple intron interruptions, but in most cases are ineffective for aligning very long sequences. Furthermore, when performing the pairwise alignment step between two very long sequences, such as mtDNA (Iborra *et al.*, 2004), or whole genomes, many algorithms either run out of memory or take too long to complete (Delcher *et al.*, 1999). To the best of our knowledge, there is no efficient and effective pairwise alignment method that requires O(n) time complexity.

读摘要给我的感觉是将原MSA问题归为成对的序列校准是采用center策略的优势，但是此处说对all都要这一步，所以我觉得有点奇怪，摘要是否应该突出真正的优势。

3. Algorithms

不是只用了star吗?

MASC uses a combination of three ideas: First, a pairwise alignment algorithm was developed, with time complexity O(*n*), based on a powerful data structure suffix tree. Then our new, center-star strategy algorithm is used for MSA, which decreases the time cost to O(*mn*), where *m* is the number of sequences and *n* is the average length of the sequences. Finally, MASC is implemented on the Spark distributed parallel framework.

这段给我的信息是首先基于后缀树改进了成对校对算法然后再使用center star策略，但是摘要是不这样的。。

最后，被实现在spark架构上，所以给我的感觉就是first，then，finally是将算法的先后顺序的。

3.1. Suffix Tree Pairwise Alignment

A suffix tree is a compressed trie (Aho and Corasick, 1975) containing all the suffixes of a given text as keys, and positions in the text as values. Suffix trees allow particularly fast implementations of many important string operations (Baeza-Yates and Gonnet, 1996). The suffix tree for the string S of length *n* is defined as a tree as follows:

Definition:

The tree has exactly *n* leaves numbered from 1 to *n*. Except for the root, every internal node has at least two children. Each edge is labeled with a non-empty substring of S. No two edges starting out of a node can have string-labels beginning with the same character. The string obtained by concatenating all the string-labels found on the path from the root to leaf *i* spells out suffix S[*i*..*n*], for *i* from 1 to *n*.

Ukkonen (Ukkonen, 1995) developed a new algorithm that can construct a suffix tree in linear time with O(*n*) computational space from a string S, where *n* is the

length of S. Ukkonen's algorithm also reduces suffix tree construction to O(*n*) time, for constant-size alphabets, and O(*n*log*n*) in general. Within a bioinformatics context, the tree alphabets consist of {A, G, C, T} or {A, G, C, U} for DNA or RNA, respectively, or the 20 amino acid symbols for proteins. Therefore, the time cost is linear for all molecular biological sequences (Barsky *et al.*, 2008).

As with running a pairwise alignment of sequence A and sequence B, it is assumed that the sequences to be compared are highly similar. Therefore, there are many common substrings in both A and B. The alignment process consists of the following steps, which are shown in Figure 1.

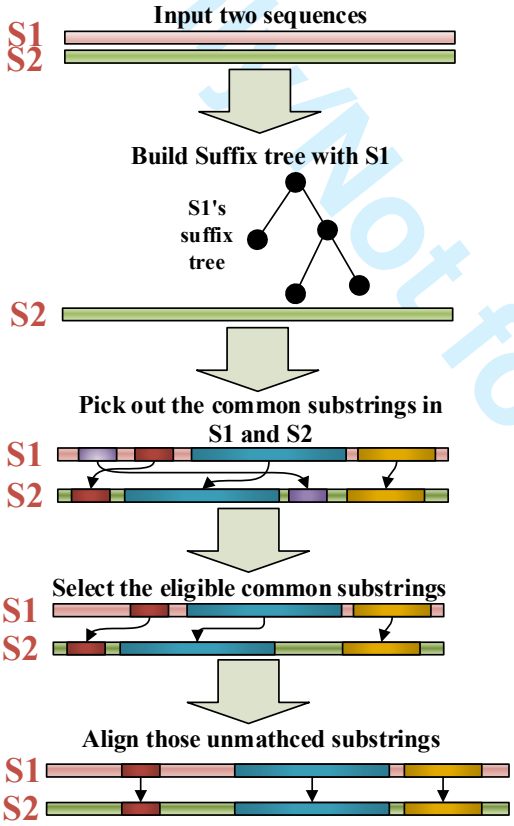


Figure 1 Pairwise alignment process using a suffix tree

First, a suffix tree is built from one sequence, here S1 is assumed as the chosen one, and the tree's name is tree-S1. Then the process of picking out common strings in S1 and S2 with sequence S2 and tree-S1 is run to form two common substring sets, set sub-S1 and set sub-S2. The next step drops out ineligible substrings in sub-S1 and sub-S2. Our standard for what constitutes eligible common substrings is described below.

后缀树和序列比较有什么优势？怎么比的?

Common substrings cannot be too short. We set the length threshold at 20 bp for nucleotide sequences. This is because very short strings are extremely common in nucleotide sequences. Very short common substrings between sub-S1 and sub-S2 would not correspond with one-to-one homology, which would lead to misalignment.

Two common substrings of sufficient length that match with each other cannot be located too remotely from one another. Here we define the position of a substring, subI, as an integer, which is subI's first character's index in string I, where subI is a substring of I. To be remote means that |position(a) − position(b)| > threshold, where a is an element of sub-S1 and b is an element of sub-S2; the threshold is set manually.

下面给的例子是reversed的，so？？？

Any two pairs of matched substrings cannot reverse positions. This means that (position(a) − position(a′)) × (position(b) − position(b′)) < 0, where a and a′ are matched substrings, and b and b′ are matched substrings. This situation is diagramed below in Figure 2.

Figure 2 Two reversed matched pairs

Next, the process of aligning the remaining unmatched substrings between S1 and S2 is run. Finally, all substrings are concatenated to form the aligned sequences.

The construction of the suffix tree can be accomplished in O(*n*) time where *n* is the length of the sequence. Picking and examining common substrings can also be accomplished in O(*n*) time. The alignments among unmatched substrings can be finished in O(km^2), where *k* is the number of substrings and *m* is the maximal length. For highly similar sequences, $k \times m$ is far less than *n*, so the total time complexity is O(*n*).

有数学推导吗?

3.2. Center-Star Strategy

The center-star strategy (Zou *et al.*, 2009) is a method that can solve the MSA problem with extremely high performance. And, the strategy's performance improves with increasingly similar sequences. For a set of sequences S = $\{s_1, s_2, \ldots, s_{n-1}, s_n\}$, the center sequence S_{center} in S needs to be found, which can satisfy the following formula:

$$S_{center} = \arg\min \left\{ \sum_{j=1, j \neq i}^{n} score(s_i, s_j) \right\}$$

score最小的? 是中心序列的?

Here $score(s_i, s_j)$ is the similarity of two sequences si and sj. After the center

sequence is selected, pairwise alignments are carried out between the center and the other sequences, one by one, to generate two set of sequences centerS = $\{s_{c_1}, s_{c_2}, s_{c_3},..., s_{c_n}\}$ and S'=$\{s'_1, s'_2, s'_3,..., s'_n\}$, where centerS is the set of center consequent sequences after pairwise alignment, and the S' is the set of consequent sequences of all other sequences. Next, all the inserted spaces are summed to obtain the final multiple sequence alignment result. The center-star strategy is shown in following algorithm outline:

Input: sequence set S= $\{s_1, s_2, s_3...s_{n-1}, s_n\}$

Output: aligned sequence set S''= $\{s''_1, s''_2, s''_3...s''_{n-1}, s''_n\}$

---

1. Run pairwise alignment between every two sequences, and construct similarity matrix W.

   校准为什么可以获得相似度矩阵？之前文章讲的校准是对齐两个序列。矩阵的形式?

2. Select S_{center} using W.

3. Run pairwise alignments between the center and the other sequences to get set centerS, which is the set of the aligned center sequence, and S', which is the set of other aligned sequences.

   这里说的set是不是指中心序列和其他每个序列进行对齐都会产生一个被对齐的中心序列?
   我是根据后文这样推测的，只看到这里不是很明白

   为什么要对齐这么多次?

   这是指?

4. Sum all the inserted spaces to get the aligned center S_{center}'

5. Run pairwise alignment between S_{center}' and the other sequences again to get S'', which is the result of the MSA.

6. Output the result.

When performing MSA among a set of similar sequences, it takes O(*n*) to finish a

pairwise alignment with our suffix tree method previously illustrated. The construction of similarity matrix W can be accomplished in O(m^2n), where m is a cardinal number of sequence set S. Pairwise alignments among the center and the other sequences can be finished in O(mn) time. The total time cost is no more than O($\mathrm{n}m^2 + mn$). The process of finding the center sequence takes most of the time. When aligning similar sequences, every sequence can be regarded as an average one. This means that each sequence can be selected as a center, so that selecting a random one is a good choice, which reduces the time complexity to O(mn).

为什么？虽然比较简单，但是最好简单说明一下怎么得到的这个时间结果的

所以第一步不需要了是吗？不用构造举证矩阵W了吗

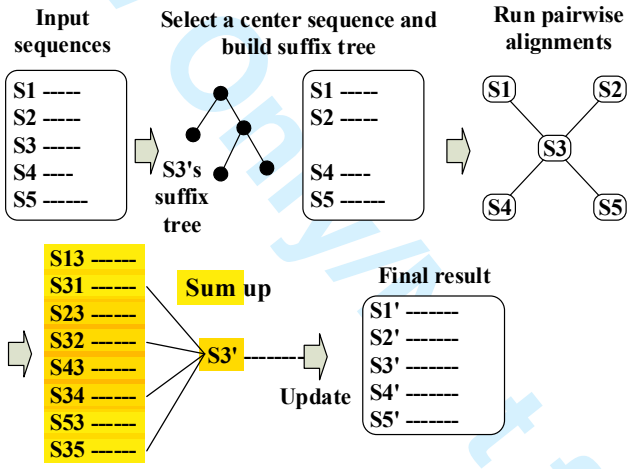
3.3 Distributed and Parallel Implementation with Spark

As MASC can perform MSA in linear time with highly similar sequences, it has enormous potential for very-high-throughput cases. We can align a larger scale of extraordinarily long sequences with better performance than other existing algorithms. However, in reality an ordinary computer cannot keep that many long sequences in memory, possibly leading to a memory crash, or wasting excessive time shifting between physical and virtual memory. Consequentially, we needed to solve the memory storage problem, and parallelize our method using the power of a distributed parallel framework.

There are many frameworks available for handling large scale data. MapReduce (Dean and Ghemawat, 2004) and Spark (Zaharia *et al.*, 2010) are the most powerful and popular. Spark is more suitable for this work, due to its memory computation characteristics.

Spark is a MapReduce type framework that can carry on all computation in memory, without needing to save intermediate results. The main abstraction Spark provides is a resilient distributed dataset (RDD), which is a collection of elements partitioned across the nodes of the cluster that can be operated on in parallel. RDDs are created by starting with a file in the file system, or an existing Scala collection in the driver program, and transforming it.

The serial version of our method is illustrated in Figure 3.



不明白这一步,s12代表什么?

Figure 3 MASC flowchart

After inputting the data, the first task is to select a random sequence as the center and build the suffix tree. Then, the pairwise alignments between the center and all other sequences are run. Next, the process of summing up the spaces among sequences is accomplished. Finally, the result is output.

? ? ?

We analyzed data flow to find 'hot spots' within the program, the processes of "pairwise alignment and sum up spaces", which occupies most of the computational

time, as shown in Figure 4.

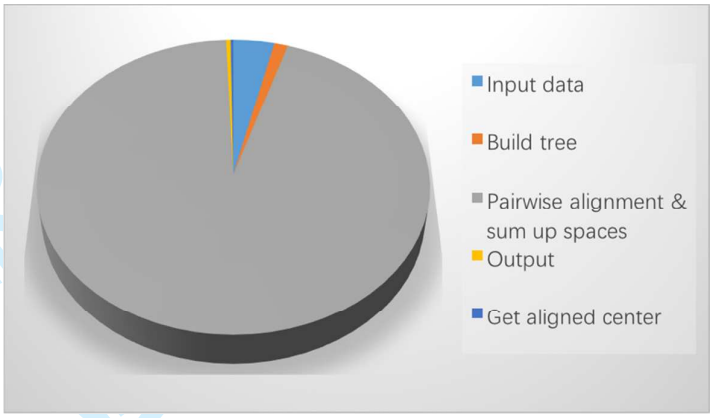
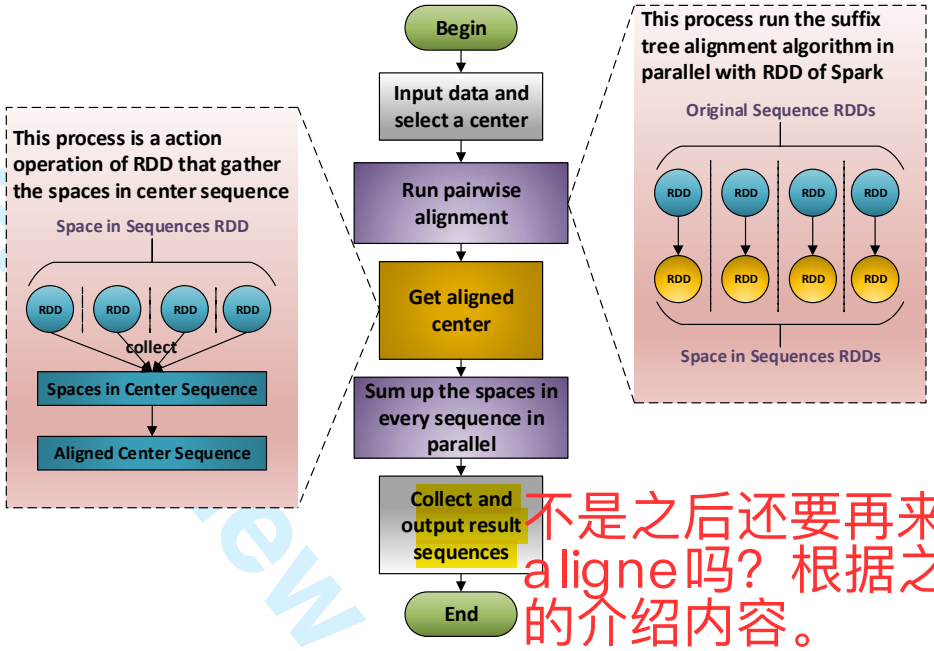


Figure 4 Time cost of MASC steps

Fortunately, in MASC the process of alignment between the center and the other sequences is readily parallelized, because there is no data dependency. Concurrently the remaining serial processes do not occupy too much time and memory. Therefore, the parallelization of MASC seemed very promising.



不是之后还要再来一次成对aligne吗？根据之前的算法的介绍内容。

Figure 5 Implementation of MASC on Spark

MASC was therefore developed within the Spark framework, and the process is shown in Figure 5. Our Spark center-star MSA has two stages. Initially, data is input from the local file system or an Apache Hadoop Distributed File System (HDFS), and a format examination is performed. Next, the sequence array is parallelized, which consists of converting the string lists to sequence string RDDs. Concurrently, the program needs to choose a center sequence, which, as previously explained, is randomly chosen, because any sequence can be regarded as average, if all are sufficiently similar. This randomly selected sequence is then used to construct a center sequence suffix tree, and pairwise alignments are run between that suffix tree and all other sequences, using Spark, making use of the algorithm previously illustrated. In this step, the suffix tree is used to get matching substrings in each sequence serially, because this step runs extremely fast, and wasting memory can be

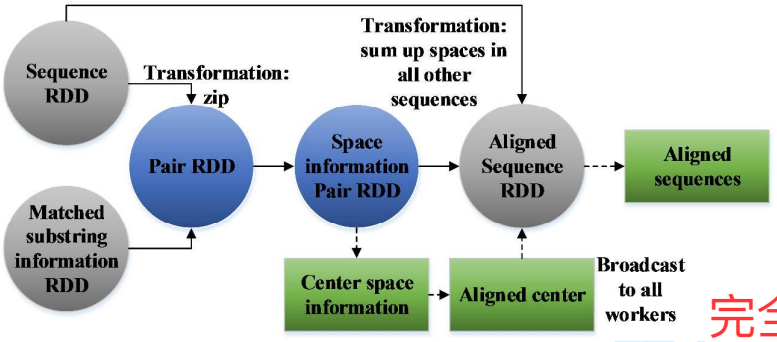
一会数组，一会集合的

怎么又出来一个list

avoided by keeping a single copy of the center suffix tree. When the information regarding all the matched substrings is obtained, a parallel process of aligning the unmatched substrings is run by passing the function to the Spark transformation, which implements the Needleman–Wunsch algorithm. In the previous step, the sequence string RDDs are transformed into pair RDDs of spaces in the center sequence and spaces in all other sequences. Next, the RDDs of spaces in the center sequence are collected to make the center aligned. Then the aligned center sequence is broadcast to all executors, and align all other sequences. In this step the pair RDDs of spaces are transformed into string RDDs of aligned sequences, and the results are stored. Finally, the aligned string RDDs are collected and are output to the local file system. The data flow and operations are shown in Figure 6.

首先我不知道space是什么，然后我也不知道为什么要转换，其次我也不明白转换的流程，图6看不懂

不是中心序列的后缀树和其他序列进行对齐吗？为什么这里是背对齐过的中心序列被广播？



完全不明白这个图。。

Figure 6 Data flow and operations

In Figure 6, all the nodes represent data and the edges represent the operations. The cycle elements are the RDDs in Spark that are distributed in the executors, the rectangle elements are datasets in the driver's memory. The operations signed by solid lines represent the transformations in Spark that transfer RDDs to subsequent RDDs and the dotted line edges represent the action operations that convert RDDs

into dataset in driver memory.

4. Experiments

4.1. Datasets and Measurements

Balibase (Thompson *et al.*, 2005) is regarded as the golden benchmark for most MSA research. However, the database is relatively small, and is only suitable for protein sequence alignment. Because there is no benchmark dataset for addressing the large-scale nucleotide MSA problem, we employed human mitochondrial genomes (mt genomes) and 16S rRNAs as the test data in our experiment. Human mt genome MSA analysis is necessary for detecting mtSNP sites, which are associated with Alzheimer's Disease, Parkinson's Disease, and Type 2 Diabetes (Tanaka *et al.*, 2004). Our human mt genome dataset contains 672 highly similar mt genome sequences, with a maximal length of 16,579 bp, and a minimal length of 16,556 bp. With the aim of testing the performance of our program with large-scale data, we duplicated the mt genomes 20 times, 50 times, and 100 times to enlarge the test set.

Table 1 Detailed information on the experimental DNA dataset

Dataset	Max length	Min length	Average length	Sequence number	File size
mt genome(1x)	16579	16556	16569.7	672	10MB
mt genome(20x)				13440	213MB
mt genome(50x)				33600	532MB
mt genome(100x)				67200	1.1GB

Our main project purpose was to accelerate the MSA process, and improve the

capacity to handle massive data. Therefore, we focused our attention on running time and throughput. We chose the sum-of-pairs (SP) value (Zou *et al.*, 2009), which is a measurement of the quality of an MSA, for measuring alignment performance. The SP value is an integer representing the sum of every pairwise alignment score from an MSA. For our purposes, in a pairwise nucleotide sequence alignment, if two nucleotides from the same column are different, one is added to the SP value, while if a space is inserted, two is added to the score, otherwise, if the two nucleotides are the same, the score remains unchanged. Thus, the SP value will be a positive integer, and the lower the SP value, the better the quality of the MSA. However, SP values are not suited for massive MSAs because the score may become too large and exceed the computer's limitations. Therefore, we employed the average SP value instead, which is the SP value divided by the number of sequences (Zou et al., 2015).

4.2. *Sequence similarity analysis*

这个标题给我的感觉是分序列的相似性，但是内容是针对不同相似程度的序列，该算法的有效性，所以不是很准确的标题

MASC is a method to do multiple sequence alignment among highly similar sequences. In order to find out what kind of sequences that are suitable to be aligned by MASC, an experiment is designed to quantify the similarity of sequences. A 1000bp long sequence is taken as basic sequence in the experiment. Then variations are carried on the copies of basic sequence. The dataset is formed by basic sequence and modified copies. The variation is to change one residue of basic sequence to an element of {A, G, C, T, -} where – means to delete the residue and concatenate to substrings. The quotient of the number of residues which are

不明白这个变换是什么意思

changed divided by length of basic sequence is called mutation rate. In the experiment, mutation rate ranges from 0 to 100%. The result shows that MASC works when the mutation rate is no more that 54% and the output is better that MAFFT measure by average SP value (Zou *et al.*, 2009). The result of comparison is shown in Figure7.

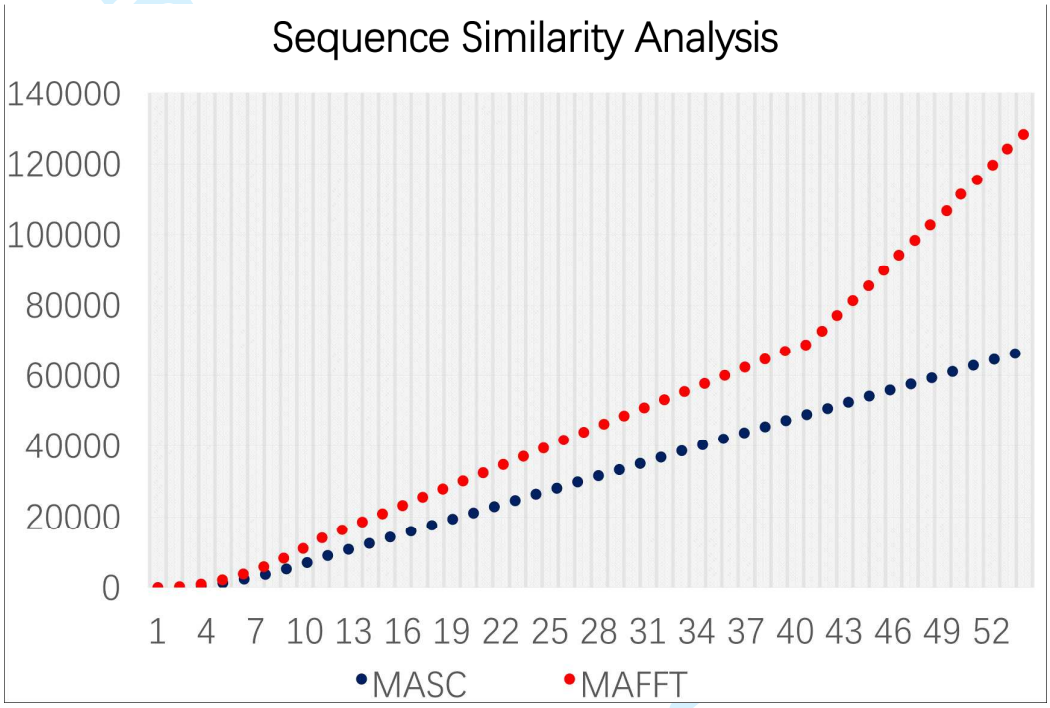


Figure 7 SP value of MASC and MAFFT with mutation rate changed

The less SP value is on behalf of better accuracy. MASC is more accurate than MAFFT when the mutation rate is no more than 54%. Due to the variation is selected from {A, G, C, T, -} in random, five elements have the same probability to be chosen, so that the residue may be unchanged. It means that the MASC works when there are more than 56.8% residues are same between sequences that need to be aligned.

4.3. Comparison with State-of-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 and KAlign. KAlign and MAFFT were run on single nodes, without any parallel operations. Therefore, for fairness, we carried out all the experiments on the same server (Intel® Xeon® CPUE7-8890v3 @ 2.5 GHz with 2 TB total memory, and the Red Hat Enterprise Linux Server release 7.1 operating system). Table 2 shows time consumption for the various human mitochondrial genome datasets.

为什么选这两个呢?

Table 2 Time consumption for different MSA tools using human mitochondrial genome datasets of different multiplicities

	10M(1x)	213M(20x)	532M(50x)	1.1G(100x)
MASC-serial	35s	10m54s	26m14s	51m51s
MASC-spark	7s	1m50s	5m11s	8m44s
MAFFT	1m59s	3h52m14s	21h54m18s	3d12h41m42s
KAlign	1h27m10s	---	---	---

From Table 2 we can see that MAFFT and KAlign take an extremely too long time to finish the MSA among long sequences, even with relatively small files. Furthermore, KAlign cannot even handle files larger than 100 MB. However, MASC (serial version) is compatible with massive files of long sequences, and the parallelized version runs extremely faster than all other programs. The parallelization analysis will be shown below.

Table 3 Accuracy of different methods

	MASC	MAFFT	KAlign
10M(1x)	14276	15202	14809

Table 3 shows a comparison of the average SP values among the different programs. As we have described previously, a lower SP value means better accuracy. Because the human mitochondrial genome dataset sequences are highly similar, the different programs perform similarly. However, the data in Table 3 shows that our method is somewhat more accurate than the other two methods with this dataset.

MASC can be used for any file, no matter how large it is. In comparison, MAFFT cannot be used for files larger than 1 GB, and KAlign cannot be used for files larger than 10MB. MASC is clearly superior for processing massive MSAs.

4.4. Speedup of Spark

We demonstrate the speedup and scalability of the Spark version of MASC in this section. We performed our experiments on a server with 16 CPUs and 2 TB of memory (same specifications as previous).

We tested the 1×, 20×, 50×, and 100× human mt genome datasets with different modes that ran various numbers of executors. Figure 7 shows the results of those experiments. Each combination of executor number and dataset was run over different 50 times.

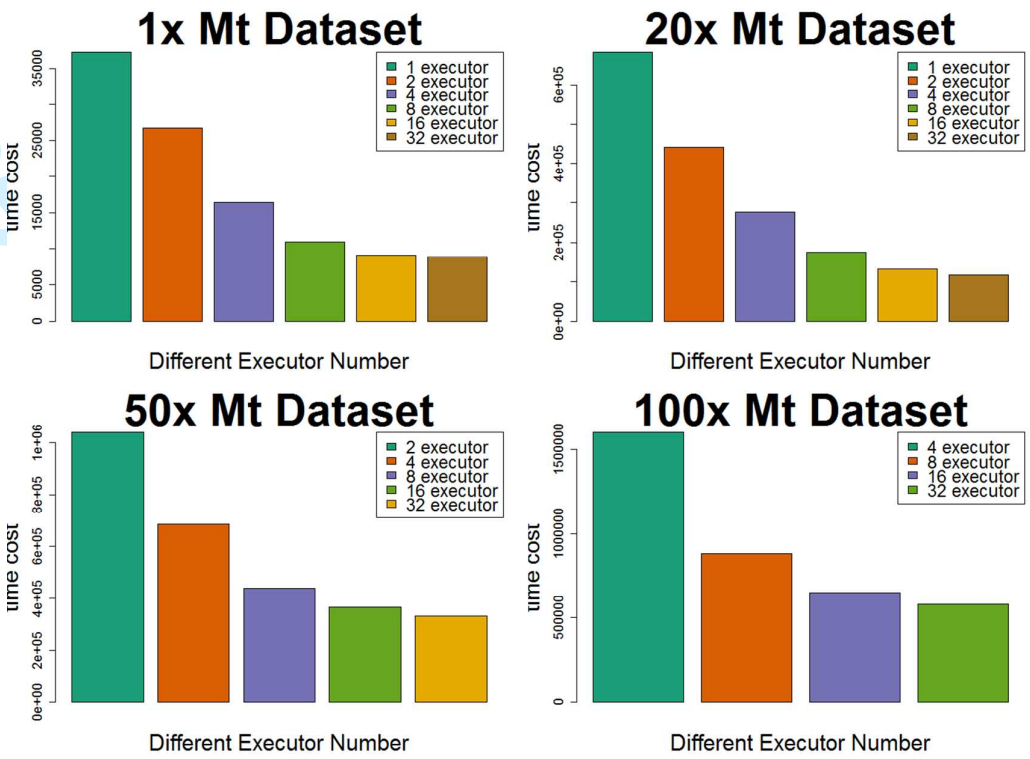


Figure 8 Running times for different mt genome datasets with different modes

In Figure 8, each bar plot shows the performance of MASC on different datasets. Each column shows the average time cost of the program running in different environments (which can be regarded as different executors on the cluster). The y-axis unit is seconds. We did not test the one and two executor modes for the 100× dataset, and the one executor mode for the 50× dataset, because it is too difficult for one node keep such a huge copy of massive data in memory. We see a remarkable speedup by Spark parallelization when we use no more than 32 workers at the same time. The speedup can be over 10 sometimes, with an average total value of eight. The scalability is well-maintained, and efficiency remains constant, while we enlarge the dataset and the number of workers at the same time.

5. Conclusion

Multiple sequence alignment is an important and fundamental bioinformatics tool. We propose MASC in this study, which can perform MSA in O(*mn*) time among highly similar sequences, where *m* is the number of sequences in the dataset, and *n* is the average length of the sequences. MASC has very high accuracy and performance. The core idea of our method is to accelerate the MSA process using three steps:

We speed up the process of pairwise alignment based on a suffix tree, which is a powerful data structure for handling strings. Time complexity is O(*n*) at the most, when aligning pairs of similar sequences based on suffix trees. A center-star strategy is then employed as a heuristic to reduce the MSA problem to pairwise alignments. MASC can be accomplished in O(*mn*) time when the sequences are highly similar. There is no loss in accuracy. Along with MASC's extremely high performance, we used the distributed parallel computing framework Spark to enlarge the memory of the system, and, in this way, the throughput of our program is substantially increased.

Extensive experiments with MASC were then performed. First, the accuracy and performance of our method was tested compared with other state-of-art tools. The scope of the comparison was quite limited, because most available tools are not optimized for performance nor efficiency. MAFFT and KAlign are two optimized tools available that were selected for comparison. The results of our experiments show that we have made great progress, and that our method has better accuracy than the

文中始终没有论述这个powerful体现在哪里

这里的then有歧义，我觉得这里的总结表述很不好如果不看前文根本看不懂

other two methods with our sample datasets, as indicated by lower average SP values.

摘要和很多概括文章的段落只说实现onSpark
但是为什么总结里出现了HDFS?

MASC has been implemented on Spark and HDFS to handle the increasingly expansive data in the field of biology and bioinformatics. The method is very suitable for parallelization because the pairwise alignments between sequences are independent and are reduced from MSA by our center-star strategy. In practice, the Spark version of MASC has great speedup and scalability. Both the single thread and the parallel tools are coded with Java, which works on multiple operating systems. Java 1.8 and Spark 2.0 are prerequisites for its operation. The codes and tools are accessible free of charge at https://github.com/suwenhecn/MASC .

Acknowledgement

The work was supported by the Natural Science Foundation of China (No. 61370010, 61272056, U1435222, 61133005) and Guangdong Provincial Department of Science and Technology under grant No. 2016B090918122.

Reference

Aho, A. V., Corasick, M.J. 1975. Efficient string matching: an aid to bibliographic search. *Commun. ACM*, **18**, 333–340.

Baeza-Yates, R. A., Gonnet, G.H. 1996. Fast text searching for regular expressions or automaton searching on tries. *J. ACM*, **43**, 915–936.

Barsky, M., Stege, U., Thomo, A., *et al.* 2008. A new method for indexing genomes using on-disk suffix trees. *Proceeding 17th ACM Conf. Inf. Knowl. Manag.*, 649–658.

Dean, J., Ghemawat, S. 2004. MapReduce: Simplified Data Processing on Large Clusters. *Proc. OSDI - Symp. Oper. Syst. Des. Implement.*, 137–149.

Delcher, A.L., Kasif, S., Fleischmann, R.D., *et al.* 1999. Alignment of whole genomes. *Nucleic Acids Res.*, **27**, 2369–2376.

Edgar, R.C. 2004. MUSCLE: a multiple sequence alignment method with reduced time and space complexity. *BMC Bioinformatics*, **5**, 113.

Gotoh, O. 1996. Significant Improvement in Accuracy of Multiple Protein Sequence Alignments by Iterative Refinement as Assessed by Reference to Structural Alignments. *J. Mol. Biol.*, **264**, 823–838.

Iborra, F.J., Kimura, H., Cook, P.R. 2004. The functional organization of mitochondrial genomes in human cells. *BMC Biol.*, **2**, 1–14.

Lounkine, E., Keiser, M.J., Whitebread, S., *et al.* 2012. Large-scale prediction and

testing of drug activity on side-effect targets. *Nature*, **486**, 361–7.

Needleman, S.B., Wunsch, C.D. 1970. A general method applicable to the search for similiarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, **48**, 443–453.

Notredame, C., Higgins, D.G., Heringa, J., *et al.* 2000. T-coffee: a novel method for fast and accurate multiple sequence alignment. *J. Mol. Biol.*, **302**, 205–217.

Smith, T.F., Waterman, M.S. 1981. Identification of common molecular subsequences. *J. Mol. Biol.*, **147**, 195–197.

Tanaka, M., Cabrera, V.M., Gonzalez, A.M., *et al.* 2004. Mitochondrial Genome Variation in Eastern Asia and the Peopling of Japan Mitochondrial Genome Variation in Eastern Asia and the Peopling of Japan. *Genome Res.*, 1832–1850.

Taylor, W.R. 1990. Hierarchical method to align large numbers of biological sequences. *Methods Enzymol.*, **183**, 456–474.

Thompson, J.D., Higgins, D.G., Gibson, T.J. 2005. BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins Struct. Funct. Genet.*, **61**, 127–136.

Thompson, J.D., Koehl, P., Ripp, R., *et al.* 1994. CLUSTAL W: Improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. *Nucleic Acids Res.*, **22**, 4673–4680.

Ukkonen, E. 1995. On-line construction of suffix trees. *Algorithmica*, **14**, 249–260.

Wang, L., Jiang, T. 1994. On the complexity of multiple sequence alignment. *J*

Comput Biol, **1**, 337–348.

Zaharia, M., Chowdhury, M., Franklin, M.J., *et al.* 2010. Spark : Cluster Computing with Working Sets. *HotCloud'10 Proc. 2nd USENIX Conf. Hot Top. cloud Comput.*, 10.

Zou, Q., Guo, M., Xiaokai, W., *et al.* (2009) An Algorithm for DNA Multiple Sequence Alignment Based on Center Star Method and Keyword Tree. *Acta Electron. Sin.*, **38**, 1746–1750.

Zou, Q., Hu, Q., Guo, M., *et al.* (2015) HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. *Bioinformatics*, **31**, 2475–2481.

Table 1 Detailed information on the experimental DNA dataset

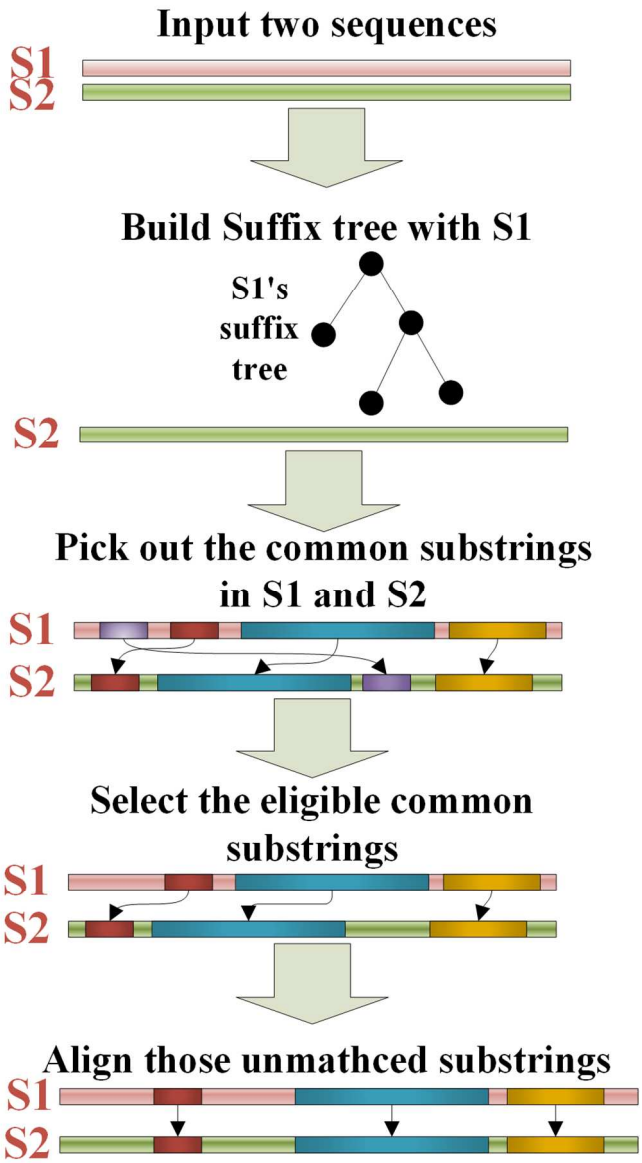
Dataset	Max length	Min length	Average length	Sequence number	File size
mt genome(1x)	16579	16556	16569.7	672	10MB
mt genome(20x)				13440	213MB
mt genome(50x)				33600	532MB
mt genome(100x)				67200	1.1GB

Table 2 Time consumption for different MSA tools using human mitochondrial genome datasets of different multiplicities

	10M(1x)	213M(20x)	532M(50x)	1.1G(100x)
MASC-serial	35s	10m54s	26m14s	51m51s
MASC-spark	7s	1m50s	5m11s	8m44s
MAFFT	1m59s	3h52m14s	21h54m18s	3d12h41m42s
KAlign	1h27m10s	---	---	---

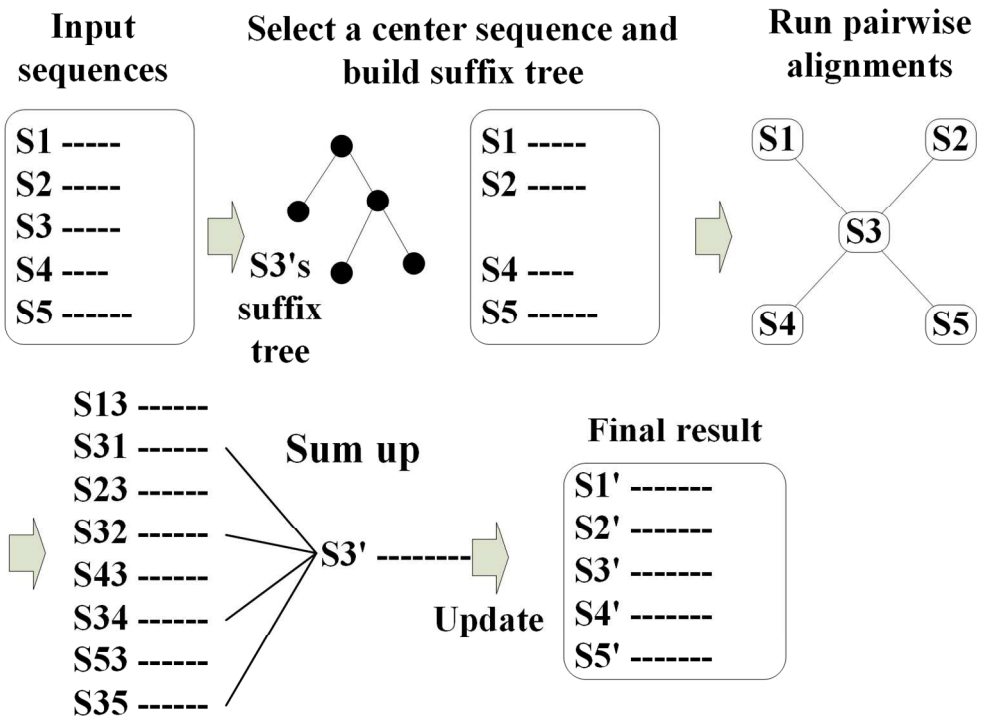
Table 3 Accuracy of different methods

	MASC	MAFFT	KAlign
10M(1x)	14276	15202	14809

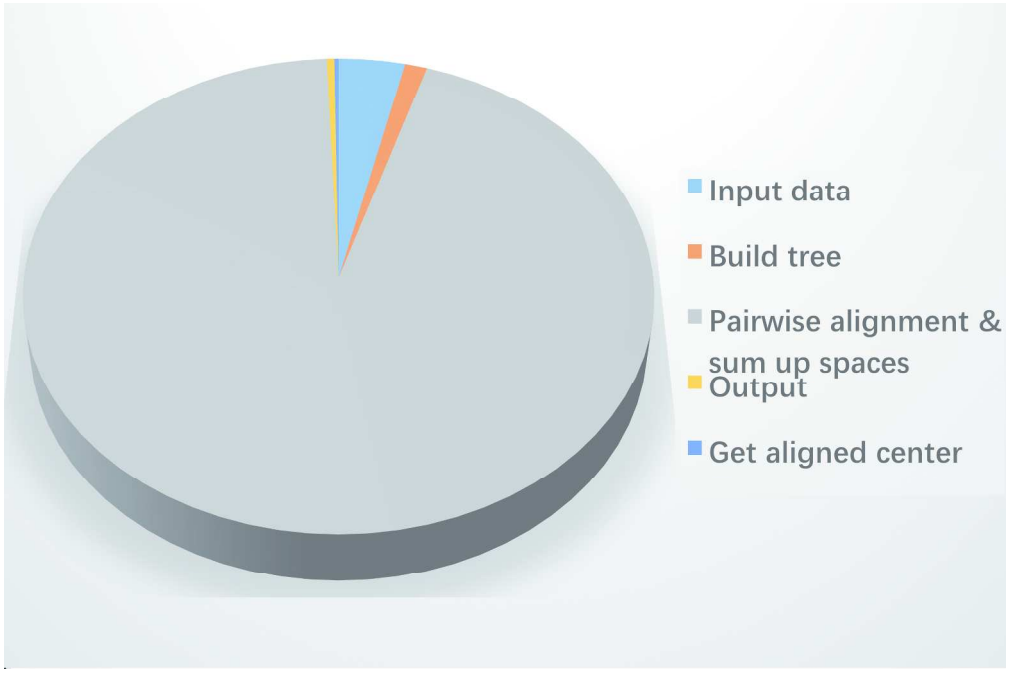


82x150mm (300 x 300 DPI)

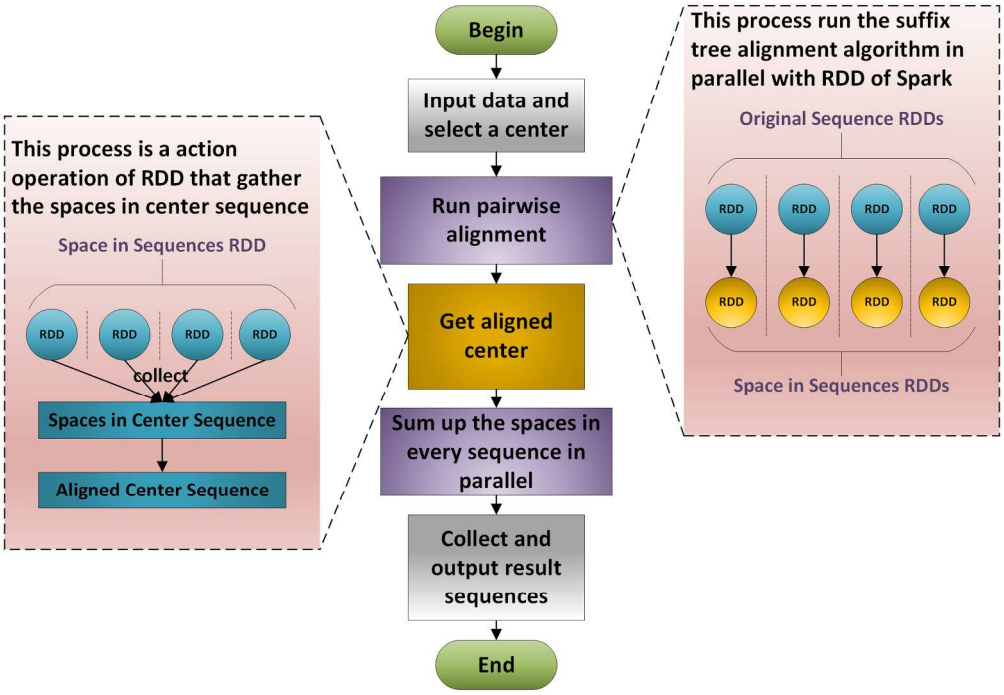
181x31mm (300 x 300 DPI)



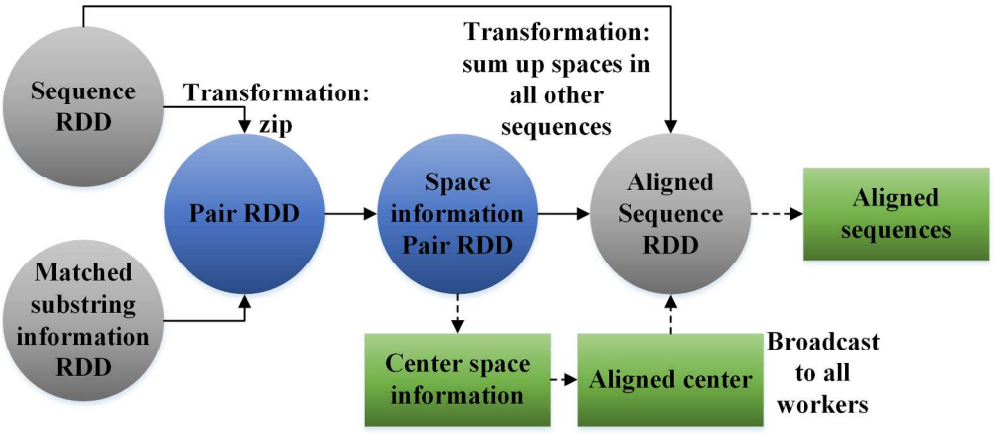
138x103mm (300 x 300 DPI)



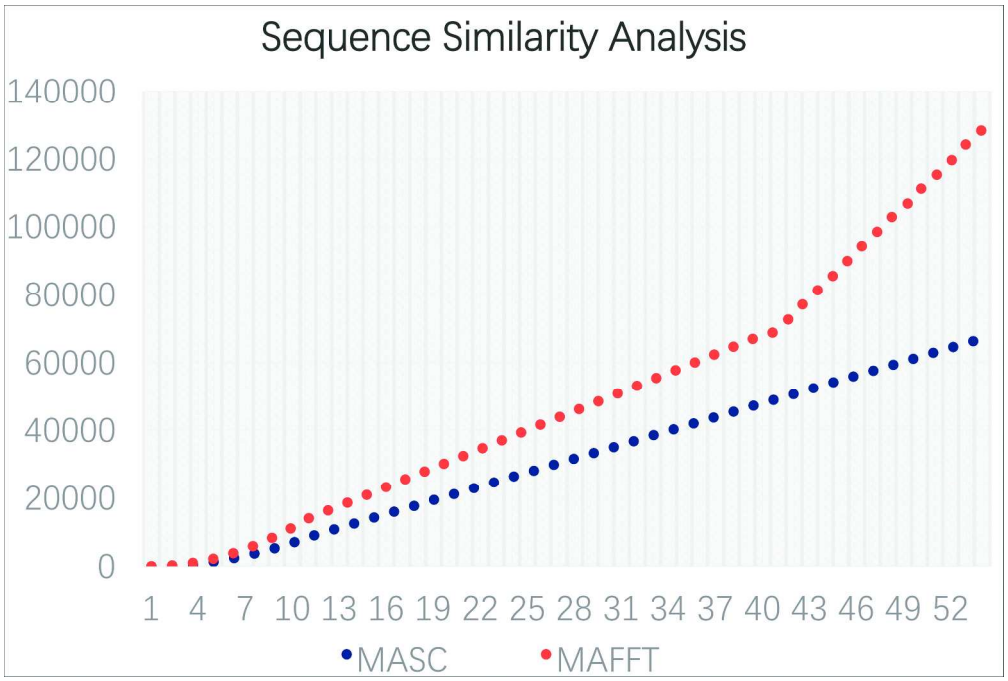
494x328mm (144 x 144 DPI)



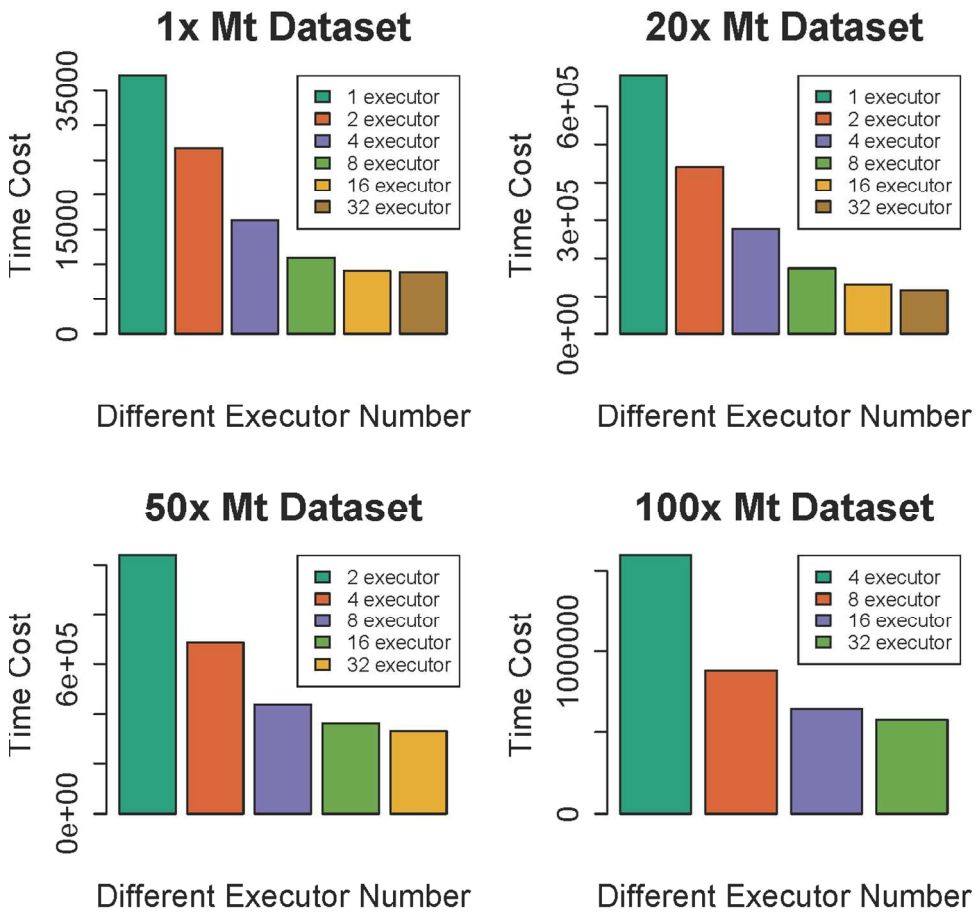
206x142mm (300 x 300 DPI)



187x80mm (300 x 300 DPI)



508x340mm (144 x 144 DPI)



177x170mm (200 x 200 DPI)