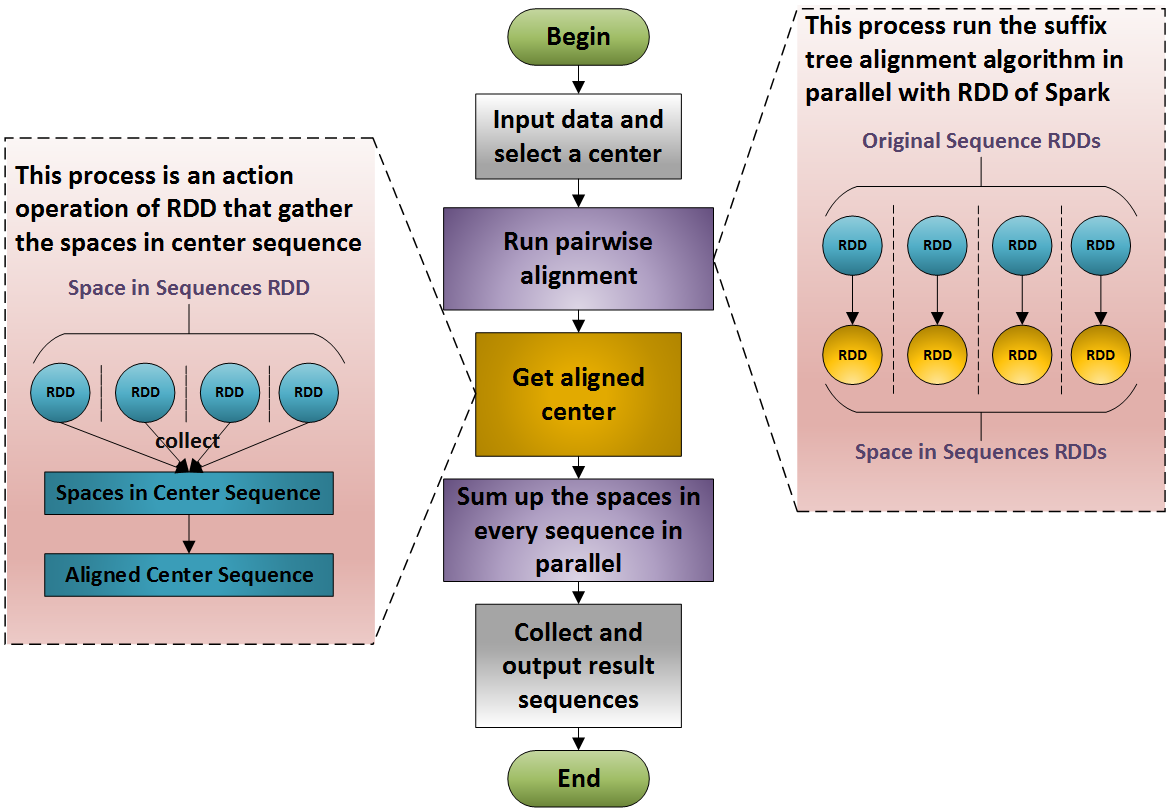
**Supplementary Method**

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

**Design of Spark version of MASC** Our Spark center-star MSA has two stages which is shown in Supplementary Figure 1.



**Supplementary Figure 1** Implementation of MASC on Spark

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 converts the string lists to sequence string RDDs. Concurrently, the program needs to choose a center sequence, which, as previously explained, is randomly chosen. This random selected sequence is then used to construct a center sequence suffix tree, and pairwise alignments are run between the tree and all other sequences. In this step, the suffix tree is used to get matching substrings in each sequence serially, because wasting memory can be avoided by keeping a single copy of the center suffix tree. When the information regarding all the matching 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 Supplementary Figure 2.



**Supplementary Figure 2** Data flow and operations

In Supplementary Figure 2, all the nodes represent data and the edges represent the operations. The cycle elements are RDDs in Spark, which 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 into subsequent RDDs and the dotted line edges represent the action operations that convert RDDs into dataset in driver memory.