# PPCAS: implementation of a Probabilistic Pairwise model for Consistency-based multiple alignment in Apache Spark

Jordi Lladós[1](✉)[0000-0001-9007-0085], Fernando Guirado[1][0000-0003-1352-9555], and Fernando Cores[1][0000-0003-2910-6709]

INSPIRES Research Center, Universitat de Lleida.
Jaume II. 69, 25001 Lleida, Spain
`{jordi.llados,f.guirado,fcores}@diei.udl.cat`

**Abstract.** Large-scale data processing techniques, currently known as Big-Data, are used to manage the huge amount of data that are generated by sequencers. Although these techniques have significant advantages, few biological applications have adopted them. In the Bioinformatic scientific area, Multiple Sequence Alignment (MSA) tools are widely applied for evolution and phylogenetic analysis, homology and domain structure prediction. Highly-rated MSA tools, such as MAFFT, ProbCons and T-Coffee (TC), use the probabilistic consistency as a prior step to the progressive alignment stage in order to improve the final accuracy. In this paper, a novel approach named PPCAS (Probabilistic Pairwise model for Consistency-based multiple alignment in Apache Spark) is presented. PPCAS is based on the MapReduce processing paradigm in order to enable large datasets to be processed with the aim of improving the performance and scalability of the original algorithm.

**Keywords:** Multiple Sequence Alignment · Consistency · Spark · MapReduce

## 1 Introduction

The probabilistic pairwise model [10] is an important step in all consistency-based MSA tools. A probabilistic model can simulate a whole class of objects, assigning an associated probability to each one. In the multiple alignment field, the objects are defined as a pair of residues from the input set of sequences, and the associated weight is the probability of being aligned [14]. For any two sequences, there are many possibilities of residue matches, $Length(sequence_1) * Length(sequence_2)$. The probabilistic model assigns each residue match a score. The higher this is, the better. For a complete dataset of sequences, the collection of the all the residue matches, which implies all the pairs of sequence evaluations, is known as the Consistency Library. This library is used to guide the progressive alignment and thus improve the final pairwise accuracy. A well-known MSA tool that uses consistency is T-Coffee [3].

The computation of the consistency library evaluates $N * (N-1)/2$ combinations, $N$ being the number of sequences, and that may be cataloged as embarrassingly parallel [3]. With the advent of the Next-Gen Sequencing, the number of sequences to align and their length have grown exponentially, with the corresponding negative impact on execution time and memory requirements. The use of massive data processing techniques can provide a solution to these limitations.

High Performance Computing (HPC) is the way to aggregate computer resources to provide parallel processing features for advanced applications. However, the fixed memory resources on each computational node and the fact that data is distributed through the interconnection network mean it is unviable for easy application to the Multiple Sequence Alignment problem. Currently, new computing technologies have been designed to manage and store huge amounts of data. These technologies, such as Hadoop [23] or Spark [17], are commonly applied to Big-Data processing and can be used to deal with this challenge. The main advantage is the ability to partition the whole data between all the nodes.

However, the increase in the number of sequences in the dataset to be treated could finally exceed the global distributed memory. The solution is the use of specialized distributed databases, such as HBase or Cassandra [1], that provide enough storage capacity to allocate any consistency library.

Thus, in the present paper, the authors present a new tool, the Probabilistic Pairwise model for Consistency-based multiple alignment in Apache Spark (PPCAS). This is able to generate the parallel probabilistic pairwise model for large datasets of proteins and can also store it in a distributed platform using the T-Coffee format.

The paper is organized as follows: Section 2 presents a brief state of the art of consistency-based MSA tools. In Section 3, we outline the development of PPCAS. In Section 4, the performance and accuracy evaluation are shown and finally, the main conclusions are presented in Section 5.

## 2 State of art

Traditional aligners, like ClustalW [7], MAFFT [6] and T-Coffee [3], are based on Gotoh [5] or Myers & Miller's [11] dynamic programming techniques, using scores from two different sources (a consistency library or substitution matrices such as PAM and BLOSUM [9]) to perform the optimal alignment of two sequences.

Unfortunately, the application of dynamic programming is inefficient for alignments consisting of many (10 - 100) sequences. Instead, a variety of heuristic strategies have been proposed, the most popular, progressive alignment [12], builds up a final alignment by combining pairwise alignments following a guide tree (beginning with the most similar sequences to the most distantly related). However, errors in the early stages not only propagate to the final alignment but may also increase the likelihood of misalignment due to incorrect conservation signals.

To lessen these early errors, consistency-based methods, such as T-Coffee[3], MAFFT[6], ProbCons[4] or DIALIGN[21], introduce consistency as a collection of pairwise alignments obtained from computing all-against-all pairwise alignments. T-Coffee uses this via a process called library extension[1]. MAFFT uses a new objective function combining the WSP score from Gotoh and the COFFEE-like score ([14]) that evaluates the consistency between multiple and pairwise alignments. ProbCons improves the traditional sum-of-pairs scoring system by incorporating Hidden Markov Models to specify the probability distribution over all alignments between a pair of sequences. Furthermore, DIALIGN-T reformulates consistency by finding ungapped local alignments via segment-to-segment comparisons that determine new weights using consistency.

The main drawback of consistency-based aligners is the high computational resources (CPU and memory) required to calculate and store the consistency information. For example, the consistency library in T-Coffee has a complexity of $O(N^2L^2)$, N being the number of sequences and L their average length. These requirements mean the method is not scalable, it being limited to aligning a few hundred sequences on a typical desktop computer. Therefore, these aligners are not feasible for large-scale alignments with thousands of sequences.

This problem of scalability is common to other tools and algorithms. Nowadays, Bioinformatics is challenged by the fact that traditional analysis tools have difficulties in processing large-scale data from high-throughput sequencing [24]. The utilization of HPC and BigData infrastructures has recently given bioinformatics researchers an opportunity to achieve scalable, efficient and reliable computing performance on Linux clusters and cloud computing services. The open-source Apache Hadoop project [23], which adopts the MapReduce framework [2] and a distributed file system, is able to store and process Petabytes of information efficiently. Moreover, Hadoop has a complete stack of services and frameworks (Spark, Cassandra, Mahout, Pig, etc) that provides a wide range of machine-learning and data-analysis tools to process any type of workflow.

Over recent years, new tools have been developed in the bioinformatics field to improve the performance and scalability of massive data processing in current applications. In [16], a novel approach is proposed that combines the dynamic programming algorithm with the computational parallelism of Hadoop data grids to improve accuracy and accelerate Multiple Sequence Alignment. In [25], the authors developed a DNA MSA tool based on trie trees to accelerate the centre star MSA strategy. It was implemented using the MapReduce distributed framework. The use of the MapReduce paradigm and Hadoop infrastructures enabled the scalability and the alignment time to be improved.

There are more MapReduce solutions in the area of mapping short reads against a reference genome. These applications, CloudBurst[18], SEAL[15] and CloudAligner[13], implement traditional algorithms like RMAP[20] and BWA[8] using the MapReduce paradigm.

[1] Given a MSA containing three sequences x, y, and z, if position $x_i$ aligns with position $z_k$ and position $z_k$ aligns with $y_j$ in the projected x-z and z-y alignments, then to be consistent the $x_i$ must align with $y_j$ in the projected x-y alignment.

## 3 PPCAS Method

The programming language selected was Python with the Ctypes extension that provides C language compatibility data types and also the ability to call external shared libraries. Thus, it is possible to obtain similar performance to native compiled code in CPU-intensive applications.

The main step in the development was to adapt the probabilistic pairwise algorithm to the MapReduce paradigm used in the big data frameworks [2]. The MapReduce paradigm enables the parallel/distributed computational resources (processors, memory and disks) to be exploited in a simple and scalable way. The MapReduce paradigm breaks down the problem into multiple Map tasks that can be executed in parallel on multiple computers/processors. After this initial Map stage, all the partial results obtained are merged and then processed by several Reduce tasks, in order to finally aggregate them.

Spark is a fast engine for large-scale data processing in real-time executed over Hadoop. Spark has a master/slave architecture. It has one central coordinator (Driver) that communicates with many distributed workers (Executors). The driver is the process where the main method runs and the executors are those that process the data received.

In the implementation of PPCAS, the map stage is responsible for defining all the tasks in charge of computing the probability score for a set of pairs of sequences. In Algorithm 1, the driver generates these tasks for all the $N * (N - 1)/2$ pair combinations (line 1) and distributes them in a balanced way among all the Map tasks using a Resilient Distributed Dataset (RDD) (line 2). Then, in line 3, the map tasks are launched and scheduled for processing on the executors. As a result, each map generates a portion of the library in parallel, and this persists in the HDFS file system.

```
   **Driver**
1: tasks_list = generate_tasks();
2: rdd_tasks = sc.parallelize(tasks_list, len(tasks_list));
3: rdd_tasks.map(executor_function).saveAsTextFile(hdfs_path);

   **Executor**
4: for each sequence $S_i \in task_i$ do
5:   for each sequence $S_j \in task_j$ do
6:     _libraryC = ctypes.CDLL("./PPCAS.so")
7:     _libraryC.pair_wise($S_i$, $S_j$)
8:   end for
9: end for
```

**Algorithm 1:** Spark parallel pairwise probability calculation

The executor, lines 4-9, performs a subset of the pairwise combinations. This is done in the double-nested loop in lines 4-5, which obtains the different combinations of sequences assigned to the *task*. It calculates the library for each of these combinations by calling the *pair_wise*($S_i$, $S_j$) function of the shared library

(PPCAS.so). This function calculates the probabilistic pairwise model for these two sequences and writes this portion of the library to the disk (HDFS).

## 4 Results and discussion

In this section we evaluate PPCAS[2]. The experimental study is focused on (1) the use of PPCAS as the main consistency library of T-Coffee by comparing the accuracy achieved and the corresponding execution time, (2) the scalability of the PPCAS when the number of nodes increases and finally, (3) the performance behavior when the number of sequences grows.

To perform the tests, we used two different multiple alignment benchmarking suites:

– BALiBASE [22] is a database of high-quality documented and manually-refined reference alignments based on 3D structural superpositions. The accuracy of the alignments is measured using two metrics: the Sum-of-Pairs (SP) and the Total Column Score (TCS), which are obtained by comparing the user alignment against a reference alignment.
– HomFam [19]: The existing benchmark datasets are very small (150 and 50 sequences in BALiBASE and Prefab respectively). Homfam provides large datasets using Pfam families with thousands of sequences. In order to validate the results of aligning a Pfam family, the Homstrad site contains some reference alignments and the corresponding Pfam family. These references are previously de-aligned and shuffled into the dataset. After the alignment process, the reference sequences are extracted and compared with the originals in Homstrad.

HomFam contains almost one hundred sets. We selected the top five manually, sorted by size (Acetyltransf, rrm, rvp, sdr and zf-CCHH) to evaluate the method. The results for the execution time presented in this section represent the average results obtained after evaluating the corresponding family. Furthermore, each experiment corresponds to five iterations in order to show the robustness of the results. The execution environment is a distributed memory cluster made up of 20 nodes, each one characterized in Table 1.

### 4.1 Evaluating the PPCAS consistency library

To assess the correctness of PPCAS, a final alignment must be done. To this end, an MSA tool is needed. TC allows the input of an externally-generated consistency library using its $-lib$ flag, so a library was built for each set with PPCAS and introduced into TC via the parameter, which generates the alignment.

This study compares the results obtained from executing T-Coffee using its own consistency library, and the same T-Coffee using the library generated with PPCAS by processing the same dataset. The experimentation focused on the differences in accuracy and the possible execution time penalties.

[2] PPCAS is available on https://github.com/jllados/PPCAS

**Table 1.** Hardware and software used in the experimentation

| Software | Version |
|---|---|
| Apache Spark | 1.6.3 |
| Apache Hadoop | 2.6 |
| Python | 2.7.13 |
| Numpy | 1.11.2 |
| GCC | 4.1.2 |

| Hardware | Model |
|---|---|
| CPU | Intel Core 2 Quad at 2.4GHz |
| RAM | 8GB DDR2 |

The BAliBASE benchmark was used for the accuracy test. The results obtained are shown in Table 2. The first column indicates the library algorithm used and the Sum-of-Pairs (SP) produced using the Bali score appears in columns 2–7. The average score over all the families is given in the last column.

**Table 2.** Comparison between T-Coffee and PPCAS library with BAliBASE.

| Library | RV11 | RV12 | RV20 | RV30 | RV40 | RV50 | Total SP |
|---|---|---|---|---|---|---|---|
| T-Coffee | 0.534 | 0.879 | 0.827 | 0.718 | 0.758 | 0.759 | 0.743 |
| PPCAS | 0.535 | 0.879 | 0.826 | 0.720 | 0.754 | 0.758 | 0.745 |

The results demonstrate that using the PPCAS library, T-Coffee is able to obtain an equivalent accuracy. The slightly differences in accuracy are due to the fact that, unlike PPCAS, T-coffee removes the smallest weighted library. This validates using the new library instead of the original one from T-Coffee.

Next, the execution time required to calculate the consistency library in T-Coffee (using the $-lib\_only$ flag) was compared with the time obtained with PPCAS, only using a single node with a quad-core processor in both cases and increasing the number of sequences. The results obtained are shown in Figure 1.

As can be observed, PPCAS always outperforms T-Coffee for execution time. However, when the number of sequences is low (100-200), the improvement is not very large, because there is not enough parallel work to obtain the maximum infrastructure performance. Nevertheless, with a large number of sequences (over 200), the PPCAS execution time improvement increases, meaning that the code is more efficient in PPCAS than in T-Coffee.

Moreover, we verified that, with 8 GB of memory, it is only possible to calculate the consistency library for a dataset with a maximum of 1,000 sequences using T-Coffee, and this takes more than 5,000 seconds. Meanwhile, PPCAS takes only 3,338 seconds to calculate the same library, which implies a 1.62x improvement. Attempts to evaluate more sequences in TC failed because the library size did not fit into the local memory.

Both the accuracy and execution time tests demonstrate that PPCAS can be used as a new method to provide the consistency library required by TC without any penalty, and furthermore, simultaneously increasing its performance.

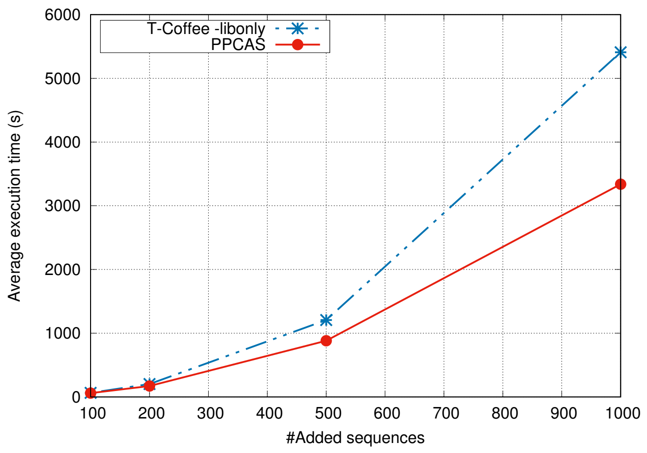

**Fig. 1.** Comparison of library building under a single node with HomFam sets.

### 4.2 Scalability study of PPCAS

To demonstrate the real benefits of using a Big-Data infrastructure, the scalability of the method when more nodes are added must be measured. We also compare the results with the original T-Coffee to have a reference point. Thus, in this test, a fixed size of 1,000 sequences (HomFam) was used, this being the maximum number of sequences TC can handle.

Figure 2 depicts the results obtained. The left axis shows the execution time, and the right one depicts the speedup obtained. It can be seen that the PPCAS speedup tends to be almost linear, taking 3,338 seconds with a single node, while it can be reduced to 183 seconds when using 20 nodes. This represents an 18.18x speedup over the single node execution time and 29.45x over the TC version presented in the previous section (5,409 seconds). These speedups are linear, denoting a good scalability as the theoretical maximum is 20x.

### 4.3 PPCAS Scalability increasing the number of sequences

This final experimentation evaluates the behavior of PPCAS with the same computational resources when the number of sequences increases.

Table 3 compares the execution time required to calculate the library in T-Coffee using a single node with a quad-core processor, (using the $-lib\_only$ parameter) with PPCAS using the complete cluster infrastructure with 20 quad-core nodes. We also analyzed the speed-up and efficiency ($speedup/nodes$, which rates the improvement against cost) as the number of sequences increases.

It is important to note that it is possible to calculate bigger libraries with PPCAS because there is no limitation to the main memory of a single node. The last column shows that the library size does not fit in the memory of a single traditional computer. Thus, it was possible to calculate the library with up to 20,000 sequences, which took 64,012 seconds.

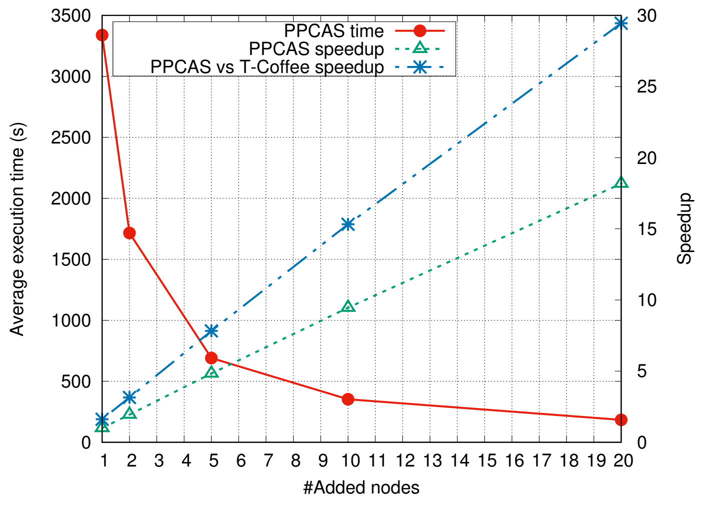

**Fig. 2.** Scalability of PPCAS with HomFam sets

When the number of sequences is low (100-200), the speedup and efficiency are not good, although the lack of parallel work mitigates the infrastructure performance. However, with a large number of sequences (more than 500), both of them achieve good values. Thus, they improve as more sequences are added.

Figure 3 shows the scalability of PPCAS on a logarithmic scale for the number of sequences to be aligned. We can observe the correlation between the size of the resulting consistency library and the time required to calculate it as the number of sequences increases. It can also be seen that the growth in execution time is proportionally smaller than the increase in size, which demonstrates the efficiency of PPCAS for calculating the library.

**Table 3.** Library building comparison between a single TC node and PPCAS multi node with HomFam sets.

| N° of seq. | T-Coffee time(s) | PPCAS time(s) | Speedup | Efficiency | Library Size (Mb) |
|---|---|---|---|---|---|
| 100 | 63.06 | 20.79 | 3.03 | 0.15 | 39 |
| 200 | 202.73 | 33.90 | 5.98 | 0.30 | 135 |
| 500 | 1,206.07 | 63.95 | 18.86 | 0.94 | 760 |
| 1,000 | 5,409.28 | 183.66 | 29.45 | 1.47 | 2,956 |
| 2,000 | — | 676.83 | — | — | 11,702 |
| 5,000 | — | 4,080.02 | — | — | 72,357 |
| 10,000 | — | 16,100.69 | — | — | 289,006 |
| 20,000 | — | 64,012.79 | — | — | 1,151,736 |

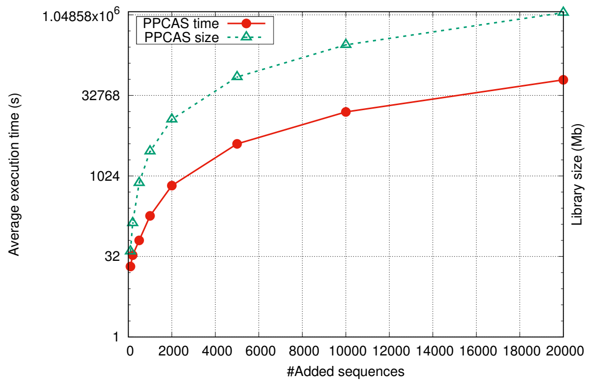

**Fig. 3.** Scalability of PPCAS regarding the execution time and output size.

## 5 Conclusions

In this paper, the authors present a scalable method to compute the probabilistic pairwise model for consistency-based multiple alignment.

We show that PPCAS is able to produce a quality library relying on a Hadoop infrastructure with Spark. In terms of execution time, the method behaves better under the same environment (single node) and benefits from almost linear speedups when more nodes are added to the ecosystem. It is also capable of computing more sequences with the same memory requirements.

In the future, we will integrate PPCAS with an aligner with a distributed database like Apache Cassandra as the interface. Storing the constraints in a high-performance database will completely eliminate the memory problems, while supplying the progressive stage with the required data. Our other aim is to reduce the execution time of the progressive itself, this being the other problematic half of an MSA with consistency.

**Acknowledgments.** This work was supported by the MEyC-Spain [contract TIN2014-53234-C2-2-R].

## References

1. Abramova, V., Bernardino, J., Furtado, P.: Which nosql database? a performance overview. *Open Journal of Databases (OJDB)* 1(2), 17-24 (2014).
2. Dean, J., Ghemawat, S.: MapReduce: A Flexible Data Processing Tool. *Communications of the ACM* 53(1), 72-77 (2010).
3. Di Tommaso, P., Moretti, S., Xenarios, I., Orobitg, M., Montanyola, A., Chang, J-M., Taly, J-F., Notredame, C.: T-Coffee: a web server for the multiple sequence alignment of protein and RNA sequences using structural information and homology extension.. *Nucleic acids research* 39(2), 13-17 (2011).
4. Do, CB., Mahabhashyam, MS., Brudno, M., Batzoglou, S.: ProbCons: Probabilistic consistency-based multiple sequence alignment. *Genome research* 15(2), 330-340 (2005).
5. Gotoh, O.: Heuristic Alignment Methods. *Multiple Sequence Alignment Methods* 29-43 (2014).

6. Katoh, K., Standley, DM.: MAFFT Multiple Sequence Alignment Software Version 7: Improvements in Performance and Usability. *Molecular Biology and Evolution* 30(4), 772-780 (2013).
7. Larkin, MA., Blackshields, G., Brown, NP., Chenna, R., McGettigan, PA., McWilliam, H., Valentin, F., Wallace, IM., Wilm, A,, Lopez, R., Thompson, JD., Gibson, TJ., Higgins, DG.: Clustal W and Clustal X version 2.0. *Bioinformatics* 23(21), 2947-2948 (2007).
8. Li, H., Durbin, R.: Fast and accurate short read alignment with Burrows–Wheeler transform. *Bioinformatics* 25(14), 1754-1760 (2009).
9. Mount, DW.: Comparison of the PAM and BLOSUM amino acid substitution matrices. *Cold Spring Harbor Protocols* 6, pdb-ip59 (2008).
10. Miyazawa, S.: A reliable sequence alignment method based on probabilities of residue correspondences. *Protein Engineering, Design and Selection* 8(10), 999-1009 (1995).
11. Myers, EW., Miller, W.: Optimal alignments in linear space. *Bioinformatics* 4(1), 11-17 (1988).
12. Nguyen, K., Guo, X., Pan, Y.: Multiple Sequences Alignment Algorithms. *Multiple Biological Sequence Alignment: Scoring Functions, Algorithms and Applications* (2016).
13. Nguyen, T., Shi, W., Ruden, D.: CloudAligner: A fast and full-featured MapReduce based tool for sequence mapping. *BMC Research Notes* 4(1), 171 (2011).
14. Notredame, C., Holm, L., Higgins, DG.: COFFEE: an objective function for multiple sequence alignments. *Bioinformatics* 14(5), 407-422 (1998).
15. Pireddu, L., Leo, S., Zanetti, G.: SEAL: a distributed short read mapping and duplicate removal tool. *Bioinformatics* 27(15), 2159-2160 (2011).
16. Sadasivam, G., Baktavatchalam, G.: A novel approach to Multiple Sequence Alignment using hadoop data grids. *International Journal of Bioinformatics Research and Applications* 6(5), 472-483 (2010).
17. Sakr, S.: Big Data Processing Stacks. *IT Professional* 19(1), 34-41 (2017).
18. Schatz, M.: CloudBurst: highly sensitive read mapping with MapReduce. *Bioinformatics* 25(11), 1363-1369 (2009).
19. Sievers, F., Dineen, D., Wilm, A., Higgins, DG.: Making automated multiple alignments of very large numbers of protein sequences. *Bioinformatics* 29(8), 989-995 (2013).
20. Smith, AD., Xuan, Z., Zhang, MQ.: Using quality scores and longer reads improves accuracy of Solexa read mapping *BMC Bioinformatics* 9(1), 128 (2008).
21. Subramanian, AR., Weyer-Menkhoff, J., Kaufmann, M., Morgenstern, B.: DIALIGN-T: An improved algorithm for segment-based multiple sequence alignment. *BMC Bioinformatics* 6(1), 66 (2005).
22. Thompson, JD., Koehl, P., Ripp, R., Poch O.: BAliBASE 3.0: Latest developments of the multiple sequence alignment benchmark. *Proteins: Structure, Function, and Bioinformatics* 61(1), 127-136 (2005).
23. Zhang, Y., Cao, T., Li, S., Tian, X., Yuan, L., Jia, H., Vasilakos, AV.: Parallel Processing Systems for Big Data: A Survey. *Proceedings of the IEEE* 104(11), 2114-2136 (2016).
24. Zou, Q.: Survey of MapReduce frame operation in bioinformatics. *Briefings in Bioinformatics.* 15(4), 637–647 (2014).
25. Zou, Q., Hu, Q., Guo, M., Wang, G.: HAlign: Fast multiple similar DNA/RNA sequence alignment based on the centre star strategy. *Bioinformatics* 31(15), 2475–2481 (2015).