INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, e-ISSN 1841-9844, 14(2), 141-153, April 2019.

Gene Sequences Parallel Alignment Model Based on Multiple Inputs and Outputs

X.L. Feng, J. Gao

Xiaolong Feng, Jing Gao*
College of Computer and Information Engineering
Inner Mongolia Agricultural University
Hohhot 010018, China
*Corresponding author: gaojing@imau.edu.cn

Abstract: Bioinformatics computing is a kind of big data processing problem, which usually has the characteristics of large data scale, large computational load and long computational time. Therefore, the use of big data technology in bioinformatics computing has gradually become a research hotspot, and using Hadoop for gene sequence alignment is one of it. It is a common way to use various tools to complete a job in the field of Biocomputing. In most studies of parallel alignment of gene sequences using Hadoop, third-party tools are also needed. However, there are few methods using Hadoop independently to complete gene sequences alignment. Adding data processing with other tools to Hadoop workflow not only affects the improvement of computing performance, but also complicates the application. In this paper, a parallel alignment model of gene sequences based on multiple inputs and outputs is proposed, which can independently complete parallel alignment of gene sequences in Hadoop platform without using other tools. This model not only simplifies the process flow of gene sequence alignment, but also improves the performance compared with other methods. This paper describes in detail the method of manipulating gene sequences with multiple inputs and outputs modes on Hadoop platform and the design of a computing model based on this method, and proves the superiority of this model through experiments.
Keywords: Multiple inputs and outputs, MapReduce, gene sequence alignment, short reads mapping, BWA (Burrows-Wheeler aligner), parallel computing.

1 Introduction

Gene sequence alignment is a time-consuming task in gene sequence analysis. With the rapid development of gene sequencing technology in terms of capacity and speed, non-parallel computing method has become a bottleneck in the flow of bioinformatics analysis. It is an urgent need to design and develop a set of stable, efficient and scalable calculation methods to solve this problem. Hadoop Distributed Parallel Computing Framework is a solution to this problem, because it provides a general method for processing large-scale data [17]. It can greatly improve the performance of large-scale data computing such as gene sequences and improve the scalability of computing methods [16]. Hadoop is a distributed computing infrastructure released by Apache Foundation. It is a high fault-tolerant and high throughput open source computing framework deployable on low-cost hardware platforms. It is very suitable for storage and computation of applications with large data sets. It provides HDFS distributed file system, YARN resource scheduling manager and MapReduce computing model [8].

BWA (Burrows-Wheeler aligner) algorithm is a gene sequence alignment algorithm widely used in bioinformatics analysis. It can map a large number of short reads gene sequences to large-scale genomes [11]. At present, mature BWA tools are single-machine serial execution or multi-threaded parallel execution, and scalable distributed parallel computing methods are being studied. Like many other bioinformatics computations, these BWA parallel computational methods require multiple tools to perform a data analysis task together. Data needs to be processed by tools before input and after output. This not only makes the calculation process cumbersome, but also affects the efficiency of data analysis. Therefore, a parallel computing model running on Hadoop framework is proposed in this paper, which can independently complete the job of gene sequence alignment. This model can be used in distributed parallel computing of BWA algorithm. In this model, the method of data multiple inputs and outputs is used, which meets the requirement of gene data operation without using other tools. Meanwhile, a MapReduce computing model matching this data input and output mode is designed, which improves the parallel computing performance of BWA algorithm. The design of this BWA parallel computing model considers the following three requirements. Firstly, the model is superior to BWA algorithm and other BWA-based parallel computing methods in performance and scalability, including BWA's own single-machine multi-threaded parallel computing method. Secondly, the results of the model should be compatible with the traditional bioinformatics analysis process. Because different versions of BWA tool contain different algorithm and characteristics, and different analysis work depends on different algorithms, so the model should provide a configurable interface to invoke the desired algorithm, instead of encapsulating only a particular algorithm. Thirdly, the model should ensure the integrity of functions. Avoid using other tools and intervention in the calculation process, and complete all tasks independently by the model. This allows users to concentrate on the scientific issues without considering the use and compatibility of multiple tools. The evaluation of this model should be compared with BWA algorithm and other parallel computing methods based on BWA algorithm in terms of performance and scalability. The advantages of the computing model are illustrated by time-consuming, speedup ratio and parallel efficiency.

2 Research background

BWA is a commonly used gene sequence alignment tool in bioinformatics analysis. It contains three algorithms: BWA-backtrack, BWA-SW and BWA-MEM. The first algorithm is suitable for sequence mapping whose length is less than 100 bp, and the latter two are suitable for sequence mapping whose length is longer. BWA-MEM is more efficient than other algorithms in sequence processing over 100 bp. The BWA mapping is a time-consuming step in gene sequence analysis, improve the efficiency of mapping has become the key to improve the biological information analysis. For this reason, BWA software also provides multithreaded parallel computing method. But this method is limited to the capacity of single-machine, and does not support distributed expansion. Therefore, its performance is not very high, especially for large genomes, which takes a long time for alignment jobs, and it may also fail because of single point failure.

Hadoop is a suitable platform for bioinformatics computing in terms of data scale, job characteristics and cost of implementation. MapReduce is a programming model suitable for handling large amounts of semi-structured data sets. The functional programming method of MapReduce is a simple way for developer. Users can implement parallel execution of computing tasks by writing the Map and Reduce functions. It provides an abstract parallel programming interface for operation, and implements the computation and processing of large-scale data in a simple way. In Map and Reduce functions, users can freely and flexibly add in parallel operation of data, which facilitates the processing of semi-structured data.

The input of BWA algorithm is usually a sequence file in FASTQ format. It is the result file of gene sequencing. The sequencing results may be single-end sequence or pair-end sequence for different sequencing methods [5, 13]. The output of BWA algorithm is SAM format file, which mainly records the location and hit times of short reads sequence mapped to reference sequence.

As to the data format, the input and output data of BWA algorithm are semi-structured data, which is suitable for MapReduce programming model. However, in the current BWA parallel computing research, there is no solution to the problem of data unified processing. Data must be processed by other tools to adapt to MapReduce programming model. If we can design a MapReduce data input and output method suitable for gene sequence processing, it will be very convenient for users to use Hadoop for bioinformatics analysis.

3 Literature review

In the research of using Hadoop to improve BWA algorithm, there are three typical representatives: BigBWA [1], Halvade [9] and SEAL [15]. BigBWA uses JNI interface to invoke BWA source code to implement distributed parallel computing of sequence alignment, which significantly improves the performance of BWA algorithm. The disadvantage is that you need to use tools to change the original format of input data before computing starts. SEAL implements MapReduce model of BWA with Python. The disadvantage is that it cannot satisfy all Hadoop native interfaces, and its running efficiency is lower than that of Java or $C++$ applications. This model only encapsulates a specific version of BWA software and cannot support the application of new versions, such as BWA-MEM for long sequence alignment. Halvade is a Hadoop-based gene sequence alignment framework developed with Java. It performs data split and sequence alignment in Map function, calls different gene analysis programs in Reduce function, and has many functions. But in data input and distribution, in order not to be restricted by Hadoop's specifications, it designed a platform-independent program "Halvade Uploader" to complete data distribution. Although multi-threading is adopted, it cannot support distributed extension and is not consistent with the Hadoop platform.

The common feature of these studies is that they all use Hadoop platform to parallelize computing tasks, which greatly improves the performance of BWA algorithm compared with serial execution. However, in these studies, third-party programs or applications independent of computing platforms are used for data preprocessing or post-processing, which makes the computing model not uniform as a whole, and also affects the improvement of computing performance to a certain extent. The reason for this approach is that Hadoop does not provide a way to directly process gene sequences, while third-party tools can easily cope with it. Taking BigBWA as an example, input data need to be preprocessed using Python program. It makes the single-end gene sequence form a single-line structure linked by $< sep >$ markers, such as $line1 < sep > line2 < sep > line3 < sep > line4$, and pair-end gene sequence form a single-line structure linked by $< part >$ markers, such as $left-end-of-sequence1 < part > right-end-of-sequence1$. At the same time, two data files of pair-end sequence are merged into one data file. The reason is that single file and single-line structure are the most convenient way for Hadoop to read directly. These additional tags need to be removed when they enter the MapReduce computing model to be accepted by the alignment algorithm. This increases the overhead of format processing in computing model. Like input data, BigBWA's output data also needs to be processed using Python programs. It merges SAM files on multiple nodes into one result file. The parallel computing model designed in this paper completes the process of data input, data distribution, distributed computing and result processing of gene sequence alignment task only with Hadoop API. In this model, input data need not be pre-processed, and can be directly input by FASTQ format single-end or pair-end sequence files. The overhead of format processing is also reduced in MapReduce computing model. Output data merging is also done without tools. Computing tasks are automatically executed without interference in the process.

4 Problem descriptions

The task of short sequence alignment is to map a large number of single-end or pair-end short reads sequences to the reference genome, and then carries out subsequent biological significance analysis. The input of alignment algorithm is short reads sequence file and reference sequence file, and the output is mapping result file. The traditional serial alignment algorithm usually takes a long time because of the huge amount of data. The principle of short sequence alignment is to find the exact position of each short reads sequence in the reference sequence by global alignment of each short reads sequence with the reference genome [12]. The principle of gene sequence alignment based on Hadoop is to distribute short reads sequence data to multiple nodes of distributed cluster, then map to reference sequence independently on each node and form their own result files. Finally, the result files on each node are aggregated to form a result file [2]. Short reads sequence is FASTQ format file. The pair-end sequence consists of left-end and right-end files. Each file consists of many short reads sequences, each of which has a fixed structure. A short reads sequence file can be regarded as a set of m sequences, and the left-end sequence can be described as $L\{l_1, l_2, l_3, ..., l_m\}$, the right-end sequence can be described as $R\{r_1, r_2, r_3, ..., r_m\}$. Then the pair-end sequence is described as $READS\{L, R\}$. Single-end sequence can be regarded as a special case with only left-end. Reference sequence is a long sequence that stores complete genetic information, REF represents reference sequence, ALN represents alignment algorithm, S represents the set of alignment results $\{s_1, s_2, s_3, ..., s_m\}$. Then the alignment problem discussed is described as

$$s_i = ALN(l_i, r_i, REF), i = 1, 2, 3...m$$

In order to distribute parallel execution of alignment tasks, $READS$ set can be divided into n subsets $D\{d_1, d_2, d_3, ..., d_n\}$ and distributed to n work nodes in the cluster. Each node will be allocated to $k = m/n$ short reads sequences if equal division of dataset is adopted. Then the data set on the work node i can be represented as

$$d_i = \{L_i, R_i\}, i = 1, 2, 3...n$$

$$L_i = \{l_{(i-1)*k+1}, L_{(i-1)*k+2}, ..., l_{i*k}\}, k = m/n$$

$$R_i = \{r_{(i-1)*k+1}, r_{(i-1)k+2}, ..., r_{i*k}\}, k = m/n$$

Computing task on node i is represented as $s_i = ALN(d_i, REF)$. Computing tasks on all nodes are executable in parallel. When the task is completed, the s_i on each node can be merged into a result file, which is compatible with the traditional biological analysis work.

To implement the distributed parallelization of alignment algorithm, four main problems need to be solved:

(1) How to input the gene sequence file directly without preprocessing?

(2) How to distribute the sequence in READS set to the work nodes?

(3) How to execute the alignment algorithm?

(4) How to merge the alignment results on the work nodes?

In data input, a single-end sequence can be regarded as a special case of a pair-end sequence. So the main problem is how to input two or more files with Hadoop APIs without preprocessing. The change of data input method will lead to the change of subsequent data calculation method, so it needs to be considered comprehensively. Hadoop provides APIs for multiple data sources to read at the same time, which can solve this problem. It also needs to design programs to meet the requirements of gene sequence operation. Gene sequences from different data sources need to be treated differently in the algorithm. It is necessary to ensure that the left-end sequence and the right-end sequence of a gene sequence can be recognized and integrated.

If there are m short reads sequences and n work nodes in an alignment job, the task of data distribution is to distribute m sequences to n work nodes. In order to get higher efficiency, the data distribution requires minimizing the amount of data movement and keep load balance on the work nodes. The simple way is to distribute m sequences in full to all work nodes. The disadvantage of this method is that it takes up a large amount of disk space, has large network traffic and takes a long time. The advantage of this method is that it does not need to design a distribution algorithm, and does not need to move data in the process of task execution. The ideal method is to distribute the necessary sequence to the designated nodes, so as to avoid moving data during task scheduling. Custom partition in Hadoop, which allows data to be distributed to designate nodes according to computing requirements, can solve the problem of data distribution very well.

In the phase of executing alignment algorithm, in order to ensure the compatibility of alignment results and the integrity of alignment function, the best way is to call the existing traditional alignment program without changing the source code. Invoking alignment program on distributed cluster can be implemented by JNI, PIPLE or SHELL. The collection and merging of result files can be accomplished on HDFS by Hadoop file manipulation.

5 Method

The goal of model design is to design a stable, reliable, efficient and scalable distributed computing model, so that short sequence alignment algorithm can be distributed and parallel implemented on Hadoop platform. The model does not need third-party tools to automate processes, including data input, data distribution, distributed computing and result collection.

The following assumptions are made for the configuration or characteristics of distributed cluster:

– Computing framework runs in distributed cluster with one name node and several data nodes with the same capacity.

– Each alignment operation can be independent of other tasks.

– The reference database is pre-deployed to the system, and all alignment tasks can be performed by any data node.

– Short reads sequence files can be split into multiple sequences and reassembled.

The function of Hadoop platform provides great convenience for designing distributed parallel model of alignment algorithm. Considering the characteristics of platform and alignment job, a computing model is designed as shown in Figure 1. HDFS is used to store input data and results, which is convenient for data distribution and sharing in Hadoop platform. Before the job submission, the pre-deployment work should be completed, that is, the reference sequence and alignment software should be deployed to each working node in advance, and the short reads sequence file should be uploaded to HDFS to facilitate the distributed deployment. In data input, Hadoop multi-input API is used to read multiple sequence files directly without data preprocessing. Two or more data files can be stored in HDFS for multiple inputs as shown in Figure 1, L and R represent two files of pair-end sequence respectively. Single-end sequence input is considered as a special case of multiple inputs. In Mapper, left-end or right-end tags are added for key-value pairs. Then the data is partitioned according to the key of the pair by using the custom partitioning algorithm, and the partitioned data is distributed to the work nodes of the cluster. In Reducer, multi-output is used to transfer the partition data to the alignment algorithm, and the alignment algorithm is invoked on the working node to implement the distributed computing of each partition. Finally, the results on each node are collected into HDFS and merged in one file.

The design of computing model needs to take advantage of the functions provided by the platform and comply with its programming specifications [20]. Therefore, according to the model shown in Figure 1, a processing flow is designed, which includes seven steps: pre-deployment, data input, Mapper processing, data partition, Merge processing, Reducer processing, and result processing [14], as shown in Figure 2.

Two classes, *FileInputFormat* and *MultipleInputs*, are provided in Hadoop API to support multiple inputs. The former uses unified Mapper processing, while the latter supports independent Mapper processing. In this paper, the *addInputPath* method of *FileInputFormat* class is used to implement multi-input. By organizing the paths of multiple data files into an array, and then passing the array as a parameter to the function, the input class can read data from multiple data sources. The input gene sequence whether left or right, forms a key-value pair with offset as key and sequence content as value.

In Mapper processing, pair-end sequences are identified as left-end or right-end and labeled, while single-end sequences are not labeled, such as Algorithm 1. The tags added in Mapper is the basis for subsequent implementation of data partition constraints and data multiple outputs. After Mapper processing, the left-end and the right-end of a pair-end sequences form key-value pairs with the same key. Because a pair-end sequence has the same offset in two files.

Algorithm 1 The Map algorithm

```
1: INPUT: $(key, value)$
2: OUTPUT: $(key, value')$
3: if $value$ stores a pair-end sequence then
4:     if $value$ stores a left-end sequence then
5:         $value'$ = addTag($value$, left-end-tag)
6:     else
7:         $value'$ = addTag($value$, right-end-tag)
8:     end if
9: else
10:     $value' = value$
11: end if
12: Context.write$(key, value')$
```

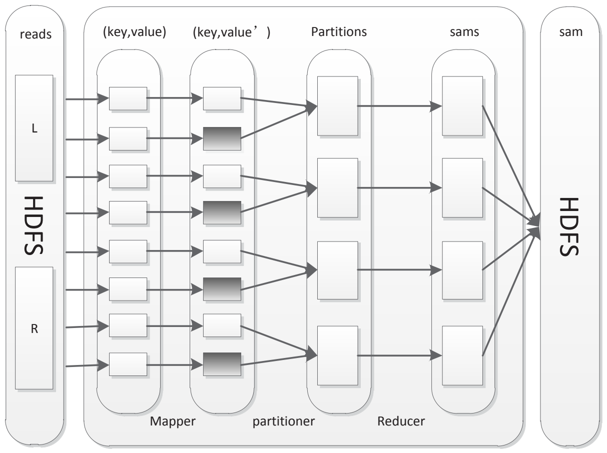

Figure 1: MapReduce model for short reads gene sequence alignment

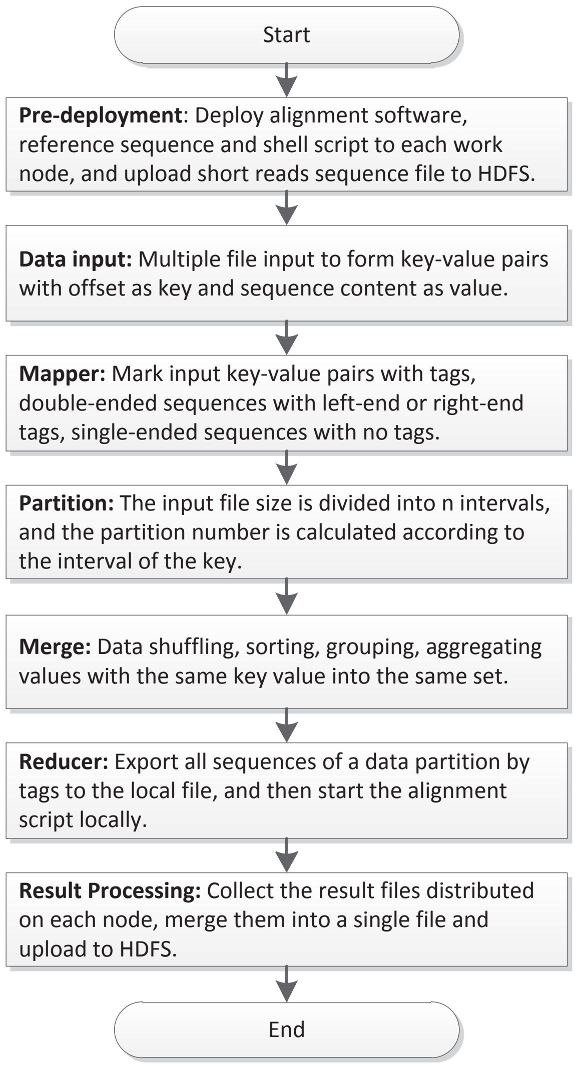
Start
Pre-deployment: Deploy alignment software, reference sequence and shell script to each work node, and upload short reads sequence file to HDFS.
Data input: Multiple file input to form key-value pairs with offset as key and sequence content as value.
Mapper: Mark input key-value pairs with tags, double-ended sequences with left-end or right-end tags, single-ended sequences with no tags.
Partition: The input file size is divided into n intervals, and the partition number is calculated according to the interval of the key.
Merge: Data shuffling, sorting, grouping, aggregating values with the same key value into the same set.
Reducer: Export all sequences of a data partition by tags to the local file, and then start the alignment script locally.
Result Processing: Collect the result files distributed on each node, merge them into a single file and upload to HDFS.
End

Figure 2: Flow chart of short reads gene sequence parallel alignment

Partitioner is a means of data distribution provided by Hadoop platform [10]. The partition classes built in the platform, such as *HashPartitioner* and *BinaryPartitioner*, are not suitable for the distribution of pair-end sequences. The operation of gene sequences has some constraints on data partitioning, which is not satisfied by Hadoop's partition classes. It is necessary to customize partition classes according to the requirements of gene sequence operation.

The constraints of data partitioning include:

- The number of left-end sequences in each partition is the same as that of right-end sequences;
- The position of sequences in a partition remains unchanged relative to that in sequence file;
- The left-end and the right-end of a pair-end sequence must be allocated to the same partition.

Since it has been assumed that the capacity of each work node in the cluster is the same, data is divided into equal partitions. The partitioning algorithm is shown in Algorithm 2. Firstly, the intervals of offsets are calculated by the size of sequence file and the number of partitions. Then partition is calculated with key in $(key, value')$. Key represents the position of a sequence in the sequence file. The partition number can be obtained by judging the offset interval of key. Sequences at the same location of the pair-end sequence have the same key, so they are assigned to the same partition.

Algorithm 2 The partition algorithm

```
1: INPUT: $(key, value')$
2: OUTPUT: $partitionNum$
3: LET $fileSize \leftarrow$ The size of input file
4: LET $partitionNum \leftarrow 0$
5: LET $n \leftarrow$ The number of nodes
6: while $partitonNum < n$ do
7:     if $key < (partitionNum + 1)fileSize/n$ then
8:         return $partitionNum$
9:     end if
10:    $partitionNum{+}{+}$
11: end while
12: return $n - 1$
```

The Merge phase will shuffle, sort, and group the data in the partition based on the key. The result of processing is that sequences with the same key are aggregated into the same set. For a pair-end sequence, a key corresponds to a set containing its left-end and right-end sequence, and the key-value pair is in the form of $(key, value'[])$.Moreover, all sequences are ordered in the partition, which ensures that the relative positions of the sequences in the partition remain unchanged. Input in reducer is a data partition and aggregated by key. If the input is a single-end sequence, the sequence in the partition will be written to the local file one by one, and if the input is a pair-end sequence, it will be identified by tags added in Mapper and output to different local files in multi-output method, as shown in Algorithm 3. Therefore, the output of Reduce function to a single-end sequence is a single local file, the content is the data partition on the node, and the output to a pair-end sequence is two local files.

After the local data file is generated, the sequence alignment task is started in the cleanup function of Reducer, and the task is executed by shell call. Shell script can be modified at any time according to the requirement of software version or parameter configuration, which makes

Algorithm 3 The reduce algorithm

```
1: INPUT: $(key, value'[])$
2: OUTPUT: $left - part_i, right - part_i$ OR $part_i$
3: for all $value$ in $value'[]$ do
4:     if $value$ stores a pair-end sequence then
5:         if $value$ stores a left-end sequence then
6:             MutiOutput.write("left-part",$value$)
7:         else
8:             MutiOutput.write("right-part",$value$)
9:         end if
10:    else
11:        Context.write($value$)
12:    end if
13: end for
```

the computing model more flexible. The alignment task is performed in parallel on each node in the cluster. Start single-end alignment algorithm for single-end sequence and pair-end alignment algorithm for pair-end sequence on the node. The output is the result of sequence alignment in the partition on the node. After the alignment is completed, the results are uploaded to HDFS. When the alignment task of all nodes has been completed, multiple result files in HDFS are merged into a unified file, and the process ends.

6 Results and discussion

6.1 Experimental design

The following experiments were designed to verify the performance of our Gene Sequence Parallel Alignment Model. The gene data were extracted from the 1000 Genome Projects [18]. The 3.3G $GRCh38.p12$ was taken as the reference genome, while two datasets, $ERR000589$ and $SRR062634$, were selected as the short reads sequence. The specific information of the sequence is shown in Table 8.

Table 1: Short reads sequence datasets

Tag	Name	Number of reads	Read length (bp)	Size(GB)
D1	NA12750/ERR000589	1.2×10^7	51	5.2
D2	HG00096/SRR062634	6.7×10^6	200	3.5

As shown in Table 8, dataset D1 is composed of pair-end sequences with a length of 51bp. It is suitable for BWA backtrack algorithm. Dataset D2 is composed of single-end sequences with a length of 200 bp. It is suitable for BWA MEM algorithm. The two datasets differ in size, sequence length, sequencing method and alignment algorithm. In order to verify the universality and stability of the computing model for different data sets, the choice of experimental data should be representative [6,7]. Therefore, two data sets with different characteristics are selected in this experiment. The test cluster is a Hadoop cluster of one name node and eight data nodes. Each node is a VMware virtual machine with 8-core CPU, 8G memory and 1T hard disk. The Hadoop uses the version of 2.7.3. The operating system is Red Hat Enterprise Linux 6.5.

- Experiment 1, the BWA mapping was performed with D1 and D2 as inputs. The same computing tasks were run on Hadoop cluster with 1, 2, 4, 6, and 8 work nodes, respectively.

The time consumption, speedup and efficiency of each task were measured to evaluate the model performance.

- Experiment 2, the BWA mapping was performed with D1 and D2 as inputs. The same computing tasks were run on single node with 1, 2, 4, 6, and 8 threads, respectively. These time-consuming are compared with those of the same tasks on distributed clusters with 1, 2, 4, 6 and 8 nodes, respectively.

6.2 Results analysis

All tasks were completed smoothly. The experimental results were the same as those of single-machine operation, but achieved at a much shorter time. Table 2 shows the time consumption, speedup and efficiency of Experiment 1. As shown in Table 2, as the number of nodes increases, the time consumption of both tasks decreases dramatically. It shows that the model has good scalability, and the experiment time can be reduced by adding more nodes. The trend of speedup ratio shows that more nodes can make the acceleration effect more obvious. But it can't achieve linear acceleration, as shown in Figure 3. The reason is that when the number of nodes in the cluster increases, the overhead for cluster management will increase, and the overhead for computing task scheduling and resource management will also increase, resulting in a decrease in resource utilization. That's why the efficiency can't always be 1 [3,4]. The dataset D1 has shorter read length and lower computational complexity. Although the total data size exceeds D2, the speedup ratio is still slightly higher than D2. It is shown that the computing model has better speedup ratio for data sets with shorter read length. Experiment 1 proves that the parallel computing model of gene sequence alignment based on multiple inputs and outputs can greatly reduce the computing time for different data sets and different alignment algorithms, and has better speedup ratio and parallel computing efficiency.

Table 2: Performance comparison of Hadoop computing model

Content	Dataset	Number of nodes				
		1	2	4	6	8
Time consumption (m)	D1	258.8	131.1	67.8	45.3	38.5
	D2	249.6	127.1	72.8	49.5	36.7
Speedup	D1	1.0	2.0	3.8	5.7	6.7
	D2	1.0	2.0	3.4	5.0	6.8
Efficiency	D1	1.00	1.00	0.95	0.95	0.84
	D2	1.00	1.00	0.85	0.83	0.85

Table 3 shows the time-consuming of Experiment 2. Generally speaking, the execution time of both methods decreases with the increase of the number of nodes or threads. Figure 4 shows the trend of time-consuming. As can be seen from the Figure 4, the BWA multithread mode slows down the time-consuming after the start of four threads. However, Hadoop computing model still maintains a good reduction in processing time after 4 nodes. Experiment 2 indicates that the performance of BWA multithread mode is affected by single-node memory and CPU capacity. It cannot increase the speed of operation by increasing the number of threads blindly, and it is not scalable. The proposed computing model has good scalability. As long as there are enough work nodes, the computing time can be reduced to a lower level. Of course, the number of nodes cannot be increased indefinitely, because increasing the number of nodes will lead to a decrease in parallel efficiency, and the balance between the number of nodes and efficiency should be achieved [25].

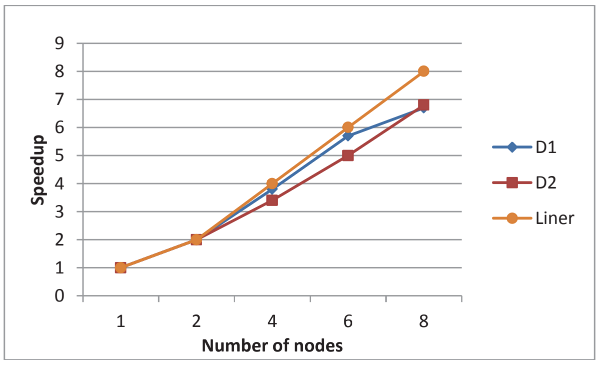

Figure 3: Speedup of Hadoop computing model with dataset D1 and D2

Table 3: Execution time of Hadoop and BWA multiple threads

Method	Dataset	Number of nodes/threads				
		1	2	4	6	8
Hadoop (m)	D1	258.8	131.1	67.8	45.3	38.5
	D2	249.6	127.1	72.8	49.5	36.7
Threads (m)	D1	258.8	154.6	127	124.6	123.8
	D2	249.6	102.5	52.9	53.2	51.8

The proposed computing model was further contrasted against several excellent parallel computing methods, which have been proved as capable of improving the performance of gene sequence alignment. All the experiments were performed using the same dataset on Hadoop clusters with different configurations. The grouping experiments were performed at 1, 2, 4, 6 and 8 nodes. Since these algorithms use different computing environments, the time consumption was not compared in the same dataset. In terms of the speedup ratio of parallel computing (Figure 5), the proposed computing model had certain advantages over the contrastive algorithms. The results show that gene sequence multiple input and output method both saves the time of data preprocessing and reduces the burden of MapReduce computing model, improving the efficiency of parallel computing.

The multiple inputs and outputs method on Hadoop platform can effectively process single-end and pair-end sequences of gene data, which makes the operation of gene sequences not limited to single file and avoids the use of third-party tools in parallel computing of gene sequence

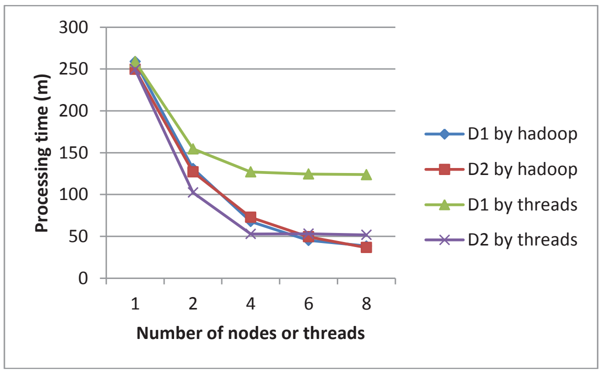

Figure 4: Execution time comparison between Hadoop and BWA multiple threads

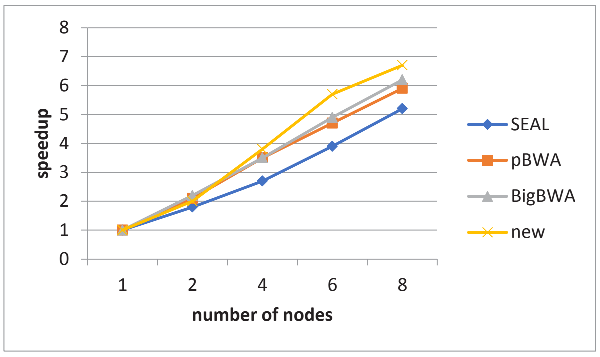

Figure 5: Speedup comparison of different parallel BWA

alignment. The parallel computing model of gene sequence alignment based on multi input and output uses Hadoop API to complete the task of gene sequence alignment, including the data input, data distribution, distributed computing and result processing, which ensures the uniformity of application. Experiments show that this computing model makes the task of gene sequence alignment scalable in distributed cluster. Compared with single node algorithm, the computing time of the same task is significantly reduced. Compared with multi-threaded parallel computing mode and other parallel gene data computing schemes, this computing model has certain advantages.

Funding

This work was supported by National Natural Science Foundation of China project 61462070.

Author contributions. Conflict of interest

The authors contributed equally to this work. The authors declare no conflict of interest.

Bibliography

[1] Abuin, J.M.; Pichel, J.C.; Pena, T.F.; Amiqo, J. (2015). BigBWA: Approaching the Burrows-Wheeler Aligner to Big Data Technologies, *Bioinformatics*, 31(24), 4003-4005, 2015.

[2] Almeida, J.S.; Gruneberg, A.; Maass, W.; Vinga, S. (2012). Fractal MapReduce decomposition of sequence alignment, *Algorithms for Molecular Biology*, 7(1), 1-12, 2012.

[3] Bala, R.J.; Govinda, R.M.; Murthy, C.S.N. (2018). Reliability analysis and failure rate evaluation of load haul dump machines using Weibull distribution analysis, *Mathematical Modelling of Engineering Problems*, 5(2), 116-122, 2018.

[4] Chen, Z.; Hou, Z.W.; Yang, Q.Q.; Chen, X.B. (2018). Adaptive Meshing Based on the Multi-level Partition of Unity and Dynamic Particle Systems for Medical Image Datasets, *International Journal Bioautomation*, 22(3), 229-238, 2018.

[5] Cock, P.J.; Fields, C.J.; Goto, N.; Heuer, M.; Rice, P.M. (2009). The Sanger FASTQ file format for sequences with quality scores and the Solexa/Illumina FASTQ variants, *Nucleic Acids Research*, 38(6), 1767-1771, 2009.

[6] Dai, Y.; Wu, W.; Zhou, H.B.; Zhang, J.; Ma, F.Y. (2018). Numerical Simulation and Optimization of Oil Jet Lubrication for Rotorcraft Meshing Gears, *International Journal of Simulation Modelling*, 17(2), 318-326, 2018.

[7] Dai, Y.; Zhu, X.; Zhou, H. ; Mao, Z. ; Wu, W. (2018). Trajectory Tracking Control for Seafloor Tracked Vehicle By Adaptive Neural-Fuzzy Inference System Algorithm, *International Journal of Computers Communications & Control*, 13(4), 465-476, 2018.

[8] Dean, J.; Ghemawat, S. (2004). MapReduce: Simplified Data Processing on Large Clusters. Proceedings of Sixth Symposium on Operating System Design and Implementation (OSD2004), *USENIX Association*, 2004.

[9] Decap, D.; Reumers, J.; Herzeel, C.; Costanza, P.; Fostier, J. (2015). Halvade: scalable sequence analysis with MapReduce, *Bioinformatics*, 31(15), 2482-2488, 2015.

[10] Gufler, B.; Augsten, N.; Reiser, A.; Kemper, A. (2012). The Partition Cost Model for Load Balancing in MapReduce, *Cloud Computing and Services Science*, Springer New York, 371-387, 2012.

[11] Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM, *Genomics*, 1-3, 2013.

[12] Li, H. (2009). The Sequence Alignment / Map (SAM) Format, *Bioinformatics*, 25(1-2), 1653-1654, 2009.

[13] Metzker, M.L. (2010). Sequencing technologies - the next generation, *Nature Reviews Genetics*, 11(1), 31-46, 2010.

[14] Pandey, R.V.; Schlotterer, C. (2013). DistMap: A Toolkit for Distributed Short Read Mapping on a Hadoop Cluster, *PLOS ONE*, 8(8), e72614, 2013.

[15] Pireddu, L.; Leo, S.; Zanetti, G. (2011). SEAL: a distributed short read mapping and duplicate removal tool, *Bioinformatics*, 27(15), 2159-2160, 2011.

[16] Schatz, M.C. (2009). CloudBurst: highly sensitive read mapping with MapReduce, *Bioinformatics*, 25(11), 1363-1369, 2009.

[17] Taylor, R.C. (2010); An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics, *Bmc Bioinformatics*, 11(S12), S1, 2010.

[18] Watson, J.D. (1990). The Human Genome Project: Past, Present, and Future, *Science*, 248(4951), 44-49, 1990.

[19] Zhang, J.; Wu, Y.Q.; Yi, H.C. (2018). Forward modelling of circular loop source and calculation of whole area apparent resistivity based on TEM, *Traitement du Signal*, 35(2), 183-198, 2018.

[20] [Online]. Available: hadoop.apache.org/, Accesed on 20 June 2018.