A compression algorithm of fastq file based on distribution characteristics analysis

***Abstract*—With the continuous development of sequencing technology scientists in the cost of DNA sequencing in reduce gradually, it also makes the number of DNA sequencing data to increase substantially .While the genome data is need to store, the traditional computer room has not enough to store such large data. Therefore, more and more genome data need to be uploaded to the cloud. Due to the speed of growth of communication have been much faster than the growth of the genomic data, so it is particularly important for genome data compression to reduce the cost of scientific research institutions and it is of great significance to speed up the sharing of genomic data. Fastq file is an important format of genomic data, and now the compression algorithm for fastq files is mainly include of DSRC, FQC, etc. These algorithms are also compressed based on the characteristics of fastq files. In order to improve the rate of compression, we propose an algorithm of DDSRC and establish the statistical models for the distribution characteristics of strings in fastq files to perform more efficient compression algorithms. This paper will explain the algorithm based on the distribution characteristics analysis and compare the results with other compression algorithms.**

Keywords—genome data; fastq; Compression algorithm; DDSRC;

I.

High-throughput DNA sequence data generated by next generation sequencing (NGS) technologies have brought tremendous stress in data storage and transmission, which poses new problems in memory and storage capabilities [1]. Most fastq file compression tools are running on a Linux platform, according to whether based on specific genetic template to compress or not, it can be divided into two kinds in general. One kind is the non-reference genome compression tool, such as FQZ_Comp, FQC and DSRC, the other is a reference genome compression tool, such as DNAZip. This project designed a common compression tool for fastq file ,and the reference genome compression algorithm facing a big problem is that there is no ready reference genome often for comparison. Due to the lack of target genome sequence which is similar to reference genome, lead to the situation that there is a great compression effect but not to be a common solution. Therefore, we adopted the compression strategy of non-reference genome, and compared with the non-reference genome algorithms of DSRC, FQZ_Comp and FQC.

The DSRC method treats the FASTQ file as a file consisting of four parts, a line as a part [2]. The four parts are: the title line, the sequence of bases in the third row and line quality, especially the third lines without compression processing. The following is the main process of the DSRC algorithm. Firstly the header row as several serial string with a delimiter, according to the characteristics of the title string, using the corresponding coding technology that can get the maximum compression including Huffman coding or differential coding. The next is the transfer of additional symbols from DNA data to quality data in addition to the four bases of ATCG. Then analyze the statistical model of statistic sequence, and according to the statistical model and the range of the quality of the base, adopt corresponding encoding, mainly including Huffman coding and coding directly .After that can be used to make code for the secondary compression. Last is the compression of line quality, according to the characteristics of the quality of line of the data, that is quality line from 0 to 40 plus the offset of the data of 33 scope, taking some processing, such as removing possible character that may appear in the end. After processing the data and reunification of Huffman encoding, or directly using the run-length coding. At the same time, the DSRC proposes a hierarchical structure, which divides the data into several record block and 512 record blocks as super blocks. These parameters can be adjusted according to specific files. And each superblock compressed by independent super block as a unit will also record the statistical information of each block it is that if you want to unzip a block, the first location to the record belongs to super block. It computes the block addresses in the superblock and read all records of sharing information, then can unzip record data in a block. The more Randomness of the DSRC for file to be written to improve the speed of compression. Therefore, to some extent at the expense of the compression rate. In general, the theory of DSRC maximum compression rate in the 5-1, sacrifice the part of the compression efficiency to improve the speed of compression.

Shengyu Liu, Haibing Chen, Lijia Peng, Baishan Wang, Hongli Wang *, Xinze Zhou
Xi'anTech University of Software, China
liu.s.y@foxmail.com

LZMA to compress, using different compression coding method to compress with target in order to get the best rate of compression . The final output three streams of data will be archived into a single compressed file.

translate into an integer data records in a file, so that we can implement that use the previous four characters to compress to an integer data accounts for only a byte, the compression rate of our method is 75%.

Figure 3: The chart of the weight

2.3

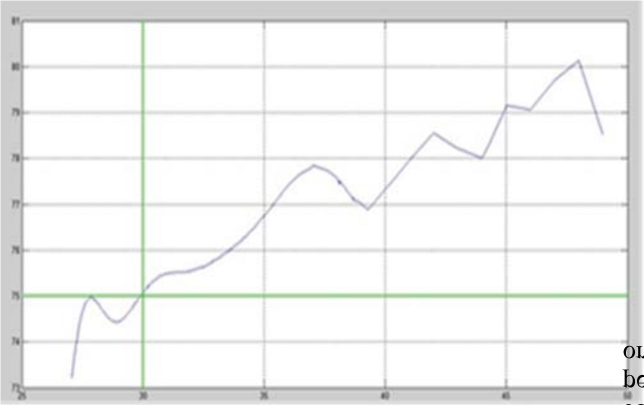
For the compression of quality score line, current compression algorithm are all very seriously about it. For quality score line, its repetitive FASTQ is very low [7]. Generally spend a lot of time to calculate the average deviation of the weight of values is poor, not a good choice .Because for quality score line, "#" is the highest probability character, so in terms of probabilistic, we can focus on the analysis of "#" character repetitive rate to balance the program running speed and compression rate.

We can see a sequence of a quality score line, and it is not particularly obvious according to rule .Generally it is a number from 0 to 40 plus the offset of 33 and get the number represented by a number of ASCII. through the observation can be found that may appear continuously someone repetitive character such as the following sequence:
<389<<<>><561,7)6?@?@##############################
###########################

This sequence is only # repeat many times, but species number and other character repeat probability is not large . If use Huffman encoding can cause excessive index record and cause compression effect to be not good, so we adopt the method of Run Length Coding to compress quality score line[8].

The Run Length Coding refers to the number of consecutive characters to replace it with the number of characters that appear. For example, AAAABBBB can be encoded as A3B3 with The Run Length Coding to reach the effect of compression. After The Run Length Coding, can continue to use 7 zip to the secondary compression to reach the higher compression rate. At the same time, in order to make the effect of the secondary compression better, we only use Run Length Coding to the characters which repeat three times or more [9].

The flow chart is as follows:



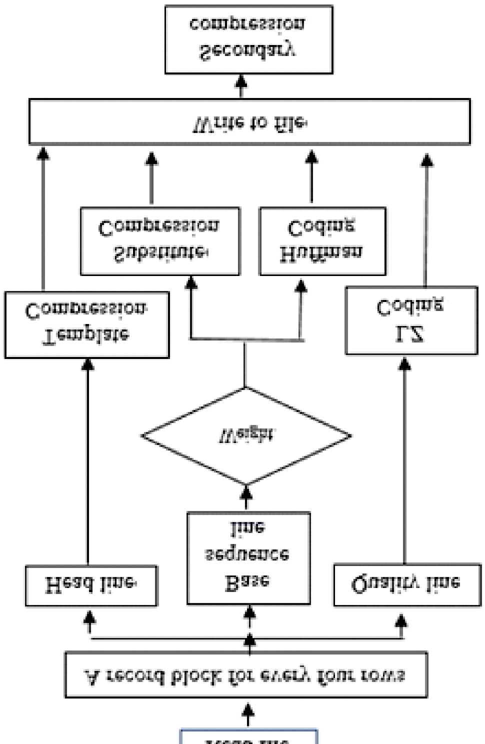


Figure4: The flow chart of the algorithm

III PROPOSED FRAMEWORK

To evaluate the performance, including compression rate, compression speed, of our algorithm, we compared against three popular compression algorithms, DSRC, FQZ and FQC. We define the rate as the result of compressed file size divide original file size. The test machine was an intel Core i5 3.1 GHz, using double core of 8G RAM, under Mac OS 10.12.6. Time measurements were performed with the Unix time command. We collected 5 files in FASTQ format for testing from1000 Genomes Project. The result of our experiment is presented in Table I.

TALEI compression result

		DSRC		FQZ		FQC		DDSRC	
Dataset	size(MB)	time(s)	rate	time(s)	rate	time(s)	rate	time(s)	rate
SRR099966	178	10	0.216	51	0.179	52	0.178	52	0.194
SRR062634	80	4	0.223	23	0.200	22	0.188	24	0.211
SRR007215_1	700	55	0.154	244	0.148	252	0.144	239	0.194
SRR003186	890	63	0.249	292	0.218	297	0.211	293	0.201
SRR003177	1200	86	0.236	401	0.207	403	0.213	390	0.216

note: the rate is equal to compressed file size divide origin file size, so lower rate means the better performance. The time means the time used to compressess the original file.

As we expected, DSRC performs very well in compress speed, it compressed faster than the other 3 algorithms in all test case, while it performs not as well as other algorithms in compression rate. The DDSRC algorithm has almost the same performance in the speed of compression when the size of origin file is less then 200MB, and as the increase of original

file size, it compresses faster than FQZ and FQC while has the similar compression rate. The implement of DDSRC has only single thread when I/O.