

PQSDC:A Novel Parallel Quality Scores Data Compressor via Sequence Partition and Run-length Prediction Mapping (Supplementary Materials)

Hui Sun, Yingfeng Zheng, Huidong Ma, Haonan Xie, Xiaofei Wang,
Cheng Zhong, Xiaoguang Liu, Gang Wang

August 7, 2023

This document presents a comprehensive guide encompassing the acquisition of the experimental dataset for the PQSDC algorithm, a detailed analysis and description of the algorithm itself, an instructive tutorial on the installation, configuration, and utilization of the PQSDC Linux toolkit, as well as supplementary experimental findings.

1 Algorithm Description and Analysis

Let k represent the value of the k-mer parameter, α represent the partition threshold factor, β represent the run-length switch factor, m represent the number of QSD sequences, n represent the average sequence length, M represent the number of samples, N represent the number of splices, Pr represent the single-node CPU core value, Q is the QSD set with compression, W is the weight vector, G is the splicing parameter, and Nu as the CPU-Cluster Nodes Number. Algorithm 1 formally expresses the quality score data parallel algorithm PQSDC based on sequence partitioning and run-length prediction models proposed in this paper.

Algorithm 1 PQSDC

Input : $k, \alpha, \beta, m, n, M, N, Pr, Q, W, G, Nu$

Output : $Q.pqsd$

BEGIN.

- 1: Parallel Initialization $B \leftarrow \{b_0, b_1, \dots, b_{m-1}\}$, $b_i \leftarrow 0$, where $i = 1, 2, \dots, m - 1$;
- 2: Initialize the k -mer frequency table $F \leftarrow \{\}$;
- 3: Initialize the $NFactor \leftarrow \text{'-INF'}$; /* -INF represents negative infinity. */

Function : QSD sequences partitioning.

- 4: Parallel traversal of the first M sequences in set Q , counting k -mers based on parameter k and populating the frequency table F ;
- 5: **for** $p = 0$ to $Pr - 1$ **do-para** /* Compute the normalization factor $NFactor$ in parallel based on equations (2) and (3). */

```

6:   for  $i = 0$  to  $M - 1$  do-para
7:     if  $p = \lceil \frac{i}{Pr} \rceil$  and  $\frac{\sum_{j=0}^{n-k+1} F[q_{i,j:j+k-1}^p]}{(n-k+1) \times M} > NFactor$  then
8:        $NFactor \leftarrow \frac{\sum_{j=0}^{n-k+1} F[q_{i,j:j+k-1}^p]}{(n-k+1) \times M}$ ; /* Mutually update  $NFactor$  */
9:     end if(line-7)
10:  end for(line-6)
11: end for(line-5)
12: for  $p = 0$  to  $Pr - 1$  do-para /* Parallel updates B via equation (1). */
13:   for  $i = 0$  to  $m - 1$  do-para
14:     if  $p = \lceil \frac{i}{Pr} \rceil$  and  $\frac{\sum_{j=0}^{n-k+1} F[q_{i,j:j+k-1}^p]}{(n-k+1) \times M} \geq \alpha$  then
15:        $b_i^p \leftarrow 1$ ;
16:     else  $b_i^p \leftarrow 0$ ;
17:     end if(line-14)
18:   end for(line-13)
19: end for(line-12)

```

Function : QSD sequences concatenation.

```

20: Initialize the  $\hat{Q}^v \leftarrow \{\}$ , where  $v = 0, 1$ ;
21: Perform parallel concatenation operation on  $Q$  based on  $N$  and  $B$ , and
    record the result in  $\hat{Q}^v$ , where  $v = 0, 1$ ;
22: Parallel initialization of set  $\hat{B}^v \leftarrow \{b_i^v\}$ , where  $b_i^v \leftarrow \{b_{i,j}^v \leftarrow \text{'NUL'}\}$ ,  $i = 0, 1, \dots, \lceil \frac{B_v}{N} \rceil - 1$ ,
     $j = 0, 1, \dots, N \times n - 1$ ,  $v = 0, 1$ ;

```

Function : QSD run-length prediction mapping.

```

23: for  $v = 0$  to  $1$  do /* Iterate over the sets  $\hat{Q}^0$  and  $\hat{Q}^1$  */
24:   for  $p = 0$  to  $Pr - 1$  do-para /* Execution of the PRMA model */
25:     for  $i = 0$  to  $\lceil \frac{B_v}{N} \rceil - 1$  do-para /* Iterate through QSD sequences */
26:       if  $\lceil \frac{i}{Pr} \rceil == p$  then
27:         Traverse the sequence  $\hat{q}_i^{p,v}$  to obtain the mode character  $\hat{C}_i^{p,v}$ 
            and the set of run-length encoded collection  $\hat{E}_i^{p,v}$ ;
28:         Calculate  $X_i^{p,v} \leftarrow \left\{ \frac{C_i^{p,v}}{n \times N}, \frac{\sum_{j=0}^{n \times N - 1} I(\hat{q}_{i,j}^{p,v} - \hat{q}_{i,j-1}^{p,v})}{n \times N - 1}, \frac{\sum_{j=0}^{n \times N - 1} (e_{i,j}^{p,v} > 3)}{n \times N}, \frac{Sor(\hat{q}_i^{p,v})}{|\Theta|} \right\}$ ;
29:         Employ  $X_i^{p,v}$  and  $W$  according to Equation (5) to obtain the
            maximum compression gain  $y_i^{p,v}$  through the proposed PRPM
            model for  $\hat{q}_{i,j}^{p,v}$ ;
30:         Initialize  $j \leftarrow 0$ ;
31:         while  $j \leq n \times N$  do
32:           if Successfully update  $\hat{b}_{i,j}^{p,v}$  and  $j$  based on  $C_i^{p,v}$ ,  $e_{i,j}^{p,v}$ ,  $y_i^{p,v}$ ,
              and  $\beta$  using equation (4) then continue;
33:           else if Successfully update  $\hat{b}_{i,j}^{p,v}$  and  $j$  based on  $C_i^{p,v}$ ,  $e_{i,j}^{p,v}$ ,
               $y_i^{p,v}$  and  $\beta$  using equation (6) then continue;
34:           else if Successfully update  $\hat{b}_{i,j}^{p,v}$  and  $j$  based on  $C_i^{p,v}$ ,  $e_{i,j}^{p,v}$ ,
               $y_i^{p,v}$  and  $\beta$  using equation (7) then continue;
35:           else Update  $\hat{b}_{i,j}^{p,v}$  and  $j$  based on  $C_i^{p,v}$ ,  $e_{i,j}^{p,v}$ ,  $y_i^{p,v}$  and  $\beta$ 
              using equation (8)
36:           end if(line-32)
37:         end while(line-31)
38:         The  $p$ -th CPU core writes the mode character  $C_i^{p,v}$  to the

```

```

39:         mapping result string  $\hat{r}_i^{p,v}$ ;
40:         for  $j = 0$  to  $n \times N - 1$  do
41:             if  $\hat{b}_{i,j}^{p,v} \geq 201$  then Write  $\hat{b}_{i,j}^{p,v}$  and  $\hat{q}_{i,j}^{p,v}$  to the string  $\hat{r}_i^{p,v}$ ;
42:             else if  $\hat{b}_{i,j}^{p,v} \neq \text{'NUL'}$  then Write  $\hat{b}_{i,j}^{p,v}$  to the string  $\hat{r}_i^{p,v}$ ;
43:             else continue;
44:             end if(line-40)
45:         end for(line-39)
46:     end if(line-26)
47: end for(line-25)
48: end for(line-23)

```

Function : Cascaded ZPAQ for final compressing via CPU-Cluster parallel.

```

49: Compute  $Bl \leftarrow \{bl_0, bl_1, \dots, bl_{2 \times G - 1}\}$  based on the block parameter  $G$  and
    the collection  $\hat{R}^v$ , where  $v = 0, 1$ ;
50: for  $nu = 0$  to  $Nu - 1$  do-para /*Cluster nodes parallel*/
51:     for  $i = 0$  to  $2 \times -1$  do-para /*Single node multi-cores parallel*/
52:         if  $\lfloor \frac{i}{Nu} \rfloor == nu$  then
53:             The  $nu$ -th node enables  $Pr$  CPU cores to use the ZPAQ algorithm
                for parallel compression of  $bl_i$ .
54:         end if(line-52)
55:     end for(line-51)
56: end for(line-50)
57: The  $Pr$  CPU cores are used in parallel to compress the partition marker
    collection  $B$  using the ZPAQ algorithm;
58: Package the compressed sets  $B$  and  $Bl$ , along with the parameter  $n$ , into a
    compressed file named  $Q.pqsdc$  using the ZPAQ algorithm.
END.

```

During the execution of Algorithm 1, the worst-case time complexity for initializing collections B , F , and parameter $NFactor$ in steps 1-3 is $O(\frac{m}{Pr})$. The time complexity for both step 4, parallel filling of the frequency table F , and steps 5-11, parallel computation of the normalization factor, is $O(\frac{M \times (n-k+1)}{Pr})$. The worst-case time complexity for step 12-19, parallel filling of set B , and step 20-21, parallel execution of sequence concatenation, is $O(\frac{M \times (n-k+1)}{Pr})$ each. The worst-case time complexity for steps 22-28, parallel initialization of set \hat{B}^v , and the traversal calculation of $C_i^{p,v}$, $E_i^{p,v}$, and $X_i^{p,v}$ is $O\left(\frac{n \times N \times \sum_{v=0}^1 \frac{B^v}{N}}{Pr}\right)$. Step 29 computes the time complexity of calculating the maximum compression gain as $O\left(\frac{\sum_{v=0}^1 \frac{B^v}{N}}{Pr}\right)$. Steps 30-37, the parallel update of result flags $\hat{b}_{i,j}^{p,v}$, has a worst-case time complexity of $O\left(\frac{4 \times n \times N \times \sum_{v=0}^1 \frac{B^v}{N}}{Pr}\right)$, while steps 38-48, the parallel computation of updated $\hat{r}_i^{p,v}$, has a worst-case time complexity of $O\left(\frac{n \times N \times \sum_{v=0}^1 \frac{B^v}{N}}{Pr}\right)$. Steps 49-59 utilize a CPU-cluster, cascading the use of the ZPAQ algorithm for compressing B and \hat{R}^v , and performing packaging operations with a worst-case time complexity of $O\left(\frac{z \times (2 \times |Bl|)}{Nu \times Pr}\right)$. Here, z represents the average execution time of the ZPAQ algorithm for serially compressing bl_i .

Because $\sum_{v=0}^1 B^v = m$ and k are constants, and $M \ll m$, the time complexity of the PQSDC algorithm is $O(\max\{\frac{M \times (n-k+1)}{Pr}, \frac{4 \times n \times N \times \sum_{v=0}^1 \frac{B^v}{N}}{Pr}, \frac{z \times (2 \times |Bl|)}{Nu \times Pr}\}) = O(\max\{\frac{m \times n}{Pr}, \frac{z \times |Bl|}{Nu \times Pr}\})$.

During the execution of Algorithm 1, the maximum space required for collections Q , \hat{Q} , and \hat{Q}^v is $O(m \times n)$. The maximum space required for set F is $O(|\Theta|^k)$, the maximum space required for set B^v is $O(\sum_{v=0}^1 B^v)$. The maximum space overhead of the algorithm ZPAQ is $O(z)$. Because both $|\Theta|$ and k are constants, the spatial complexity of PQSDC is $O(\max\{m \times n, z\})$.

2 The Baseline Algorithms and Datasets

The open source datasets SRR8386204, SRR8386224, SRR8386225, ERR7091256, ERR7091268, SRR013951, SRR027520, SRR554369, SRR17794741, SRR17794724, SRR12175235 of NCBI(<https://www.ncbi.nlm.nih.gov/sra>) database were used for experimental verification. Detailed information about the dataset is presented in Table 1. We use sra-tools(<https://github.com/ncbi/sra-tools>) to down-

Table 1: Detailed Information of the Experimental Dataset

Name	Platform	Source	SE or PE	Len (bp)	S.Number	FSize (KB)
SRR8386204	BGISEQ	M.fascicularis	PAIRED	50	39622734	7166894
SRR8386224	BGISEQ	M.fascicularis	PAIRED	50	35985720	6504957
SRR8386225	BGISEQ	M.fascicularis	PAIRED	50	37681544	6813597
ERR7091256	NextSeq	Metagenomic	SINGLE	75	41351459	9549095
ERR7091268	NextSeq	Metagenomic	SINGLE	75	42914223	9911656
SRR013951	GAIIX	H.sapiens	PAIRED	76	36424874	9482115
SRR027520	GAIIX	H.sapiens	PAIRED	76	48493370	13334955
SRR554369	GAIIX	Pseudomonas	PAIRED	100	3315742	912888
SRR17794741	MGISEQ	M.musculus	PAIRED	100	4893024	1375389
SRR17794724	MGISEQ	M.musculus	PAIRED	100	4659890	1309645
SRR12175235	NovaSeq	Metagenome	PAIRED	151	17320189	11906486
Total	—	—	—	—	219959565	72618889

load experimental datasets. For paired-end sequencing files, we use the **cat** command to merge them into a single file for experimental testing.

We also provide a download script for the aforementioned dataset. The specific execution command is as follows:

```

1 cd data
2 data_download.sh > data_download.log &

```

The experiment compares our proposed algorithm, PQSDC, with the state-of-the-art general-purpose compression algorithms, 7-Zip, PIGZ, PBzip2, ZPAQ, as well as six latest specialized lossless compression algorithms for quality scores data, namely CMIC, LCQS, AQUa, FCLQC, Qscomp, and QVZ. All algorithms were set to the optimal compression mode. Table 2 provides detailed experimental configurations for the benchmark algorithms used in this study.

Notes: The specialized compression algorithms Qscomp and QVZ support both lossless and lossy compression. However, in this study, we conducted

Table 2: Detailed Information of the Experimental Dataset

Algorithm	Parameters	Language	Source Code
7-Zip	-mx 9-mmt 28	C/C++	https://www.7-zip.org
PIGZ	-11 -p 28	C/C++	https://www.zlib.net/pigz
PBzip2	-9 -m 2000 -p 28	C/C++	https://linux.die.net/man/1/pbzip2
ZPAQ	-method 5 -thread 28	C/C++	http://mattmahoney.net/dc/zpaq.html
CMIC	-thread 28	C/C++	https://github.com/Humonex/Cmic
LCQS	-thread 28	C/C++	https://github.com/SCUT-CCNL/LCQS
AQUa	-wsize 4-cabac true	JAVA	https://github.com/tparidae/AQUa
FCLQC	-precision 35-thread 28	RUST	https://github.com/Minhyeok01/FCLQC
Qscomp	-thread 28 lossless mode	C/C++	—
QVZ	-f 1lossless mode	C/C++	https://github.com/mikelhernaez/qvz

comparative tests using the lossless compression mode. The tested versions of the general-purpose compression algorithms 7-Zip, PIGZ, PBzip2, and ZPAQ were V22.01, V2.7, V1.1.13, and V7.15, respectively.

3 Installation, Configuration, and Usage Tutorial

PQSDC is a freely available QSD compression software. First, install and configure the PQSDC toolkit using the following command:

```

1 # Firstly, clone our tools from GitHub:
2 git clone https://github.com/fahaihi/PQSDC.git
3 # Secondly, turn to PQSDC directory
4 cd PQSDC/pqsdv_v2
5 # Thirdly, Run the following command(Warning!:GNU Make > 3.82.)
6 bash install.sh
7 # Finally, Configure the environment variables :
8 export PATH=$PATH:pwd/
9 export PQSDC_V2_PATH="pwd/"
10 source ~/.bashrc

```

The PQSDC toolkit includes two operating modes: Basic and Advanced. In order to facilitate usage on any PC, we have currently only open-sourced the single-node parallel algorithm in the current version.

```

1 Basic Usage: pqsdv_v2 [command option]
2   -c [qualities file] [threads]           #compression.
3   -d [pqsdv generate directory] [threads] #decompression.
4   -h                                     #help message.
5 Advanced Usage:pqsdv_tools [command option]
6   -fileinfo [input-fastq-file]           #statistic-infor.
7   -dirinfo [input-dir-name]             #statistic-infor.
8   -verify [source-fastq-file] <mode> [verify-file] #verify.
9     <mode> = reads
10    <mode> = qualities
11   -filesplite [input-fastq-file] mode <mode> #splite FastQ-file.

```

```

12         <mode> = ids
13         <mode> = reads
14         <mode> = describes
15         <mode> = qualities
16         <mode> = all

```

We present the validation dataset at **PQSDC/data/test.qualities**. Here are some examples of using PQSDC:

1): Using 8 CPU cores for compression.

```

1  cd ${PQSDC_V2_PATH}data
2  pqsd_v2 -c test.qualities 8
3  # Results
4  compression mode.
5  fileName : test.qualities
6  threads : 8
7  savepath : test.qualities.partition/result.pqsd_v2
8  -----
9  1 reads partition, generate test.qualities.partition directory.
10 2 parallel run-length encoding prediction mapping.
11 3 cascade zpaq compressor.
12 4 pacing files into test.qualities.partition/result.pqsd_v2.
13 5 removing redundant files.
14 over!
15 -----

```

2): Using 8 CPU cores for decompression.

```

1  pqsd_v2 -d test.qualities.partition 8
2  # Results
3  running pqsd algorithm at Sat Jun 17 15:31:22 CST 2023
4  de-compression mode
5  fileName : test.qualities.partition
6  threads : 8
7  savepath : test.qualities.partition.partition.pqsd_v2
8  -----
9  1 unpacking test.qualities.partition/result.pqsd_v2.
10 2 unsing zpaq decompression files.
11 3 parallel run-length encoding prediction mapping.
12 4 merge partitions to restore the original file
13 over
14 -----

```

3): Verify if the decompression is successful.

```

1  pqsd_tools -verify test.fastq qualities test.qualities.pqsd_de_v2
2  # Results
3  lossless recover all qualities.

```
