Optimizing Large-Scale Genomic Sequencing Reads Compression via Memory Modeling and Redundant Clustering: Supplementary Data

Hui Sun, Yingfeng Zheng, Xiaoguang Liu , Gang Wang

School of Computer Science, Nankai University, Tianjin 300000, China,

December 28, 2022

This document provides instructions for theoretical analysis, optimization method details, installing and running the optimization compression tool, downloading datasets, and some additional results. The proposed optimizing method PMFFRC and related instructions can be found at https://github.com/fahaihi/PMFFRC.

S1. Entropy Theoretical Analysis

Let $F = \{F_0, F_1, \dots F_{v\text{-}1}\}$ denotes a group of fastq files, $R_i = \{R_0^i, R_1^i, R_2^i, \dots, R_{(|F_i|/4)-1}^i\}$ denotes a collection of string reads in the i-th fastq sequencing file, where $i = 0,1,2,\dots,v$. The design opinion of reference-free reads compressors is to employ the information of the reads collections themselves to replace redundant details effectively (Chandak *et al.*, 2018; Tang and Li, 2021). From the perspective of the entropy hypothesis, the information entropy $H(X_i)$ of the initial reads collection R_i, the information entropy $H(Y_i)$ after redundancy substitute, and the redundant information conditional entropy $H(X_i \mid Y_i)$ satisfy:

$$H(X_i) = H(Y_i) + H(X_i \mid Y_i) \tag{1}$$

The reference-free based compressors begin from the redundant information of the sequencing reads collection R_i themself, maximizes $H(X_i \mid Y_i)$, and minimizes $H(Y_i)$, where $i = 0,1,2,\dots,v$.

First consider the joint compression of two fastq files. If sequencing reads in any fastq files F_a and F_b are jointly compressed, for their reads collection R_a and R_b, the total information entropy $H(Y_{a,b})$ after redundancy replacement satisfies:

$$H(Y_{a,b}) = H(Y_a) + H(Y_b) - H(Y_a;Y_b) \tag{2}$$

In formula (2), $H(Y_a;Y_b)$ denotes the mutual information entropy of the joint compressed files F_a and F_b after redundant replacement, where $H(Y_a;Y_b) \geq 0$. Therefore, formula (2) can be written as:

$$H(Y_{a,b}) \leq H(Y_a) + H(Y_b) \tag{3}$$

Then consider the joint compression of three fastq files. For the reads sets R_a, R_b and R_c, formula (4) can be obtained according to formula (1) and formula (2):

$$H(Y_{a,b,c}) = H(Y_a) + H(Y_b) + H(Y_c) - H(Y_a;Y_b) - H(Y_a;Y_c) - H(Y_b;Y_c) + H(Y_a;Y_b;Y_c) \quad (4)$$
$$= H(Y_{a,b}) + H(Y_c) - H(Y_a,Y_c) - H(Y_b,Y_c) + H(Y_a;Y_b;Y_c)$$

In formula (4), $H(Y_a,Y_c) \geq 0$, $H(Y_b,Y_c) \geq 0$, and $H(Y_a,Y_c) + H(Y_b,Y_c) \geq H(Y_a; Y_b; Y_c)$, thus, formula (4) satisfies:

$$H(Y_{a,b,c}) \leq H(Y_{a,b}) + H(Y_c) \leq H(Y_a) + H(Y_b) + H(Y_c) \quad (5)$$

Therefore, for the joint compression of v files, we can obtain that the information entropy after the redundant replacement of the joint compression of v files satisfies:

$$H(Y_{0,1,2,\ldots,v\text{-}1}) \leq H(Y_0) + H(Y_1) + H(Y_1) + ,\ldots, + H(Y_{v\text{-}1}) \quad (6)$$

Established on the above entropy insights, we get the following conclusions: For multiple fastq files joint compression, if memory usage is unlimited, the total information entropy of jointly compressing v fastq files will decrease as v increases. Therefore, the less information entropy after joint compression, the fewer bits are required to encode DNA characters. However, the system memory of the compression server is limited, so multiple fastq files need to be compressed in batches under a safe memory threshold to improve the robustness of cascaded reads compressors.

S2. Sequencing files clustering

Let $|F_i|$ denotes the number of lines in F_i, n denotes the average length of sequencing reads, Y represents the cascaded optimization compressor, U_{ram} represents the user-preset safe memory threshold (less than system memory), and K represents the clustering parameter, where $i = 0,1,2,\ldots,v\text{-}1$.

The PMFFRC algorithm achieves high compression ratios by clustering redundant reads together in different fastq files and increases cascaded compressor robustness by memory modeling. In the PMFFRC workflow, sequencing files clustering includes four stages: pre-compression, feature extraction, similarity calculation, and fastq files clustering.

Pre-compression

The purpose of pre-compression is to evaluate the maximum memory consumption of the compressed fastq format dataset F. Let Y_{res} denotes the resident memory of compressor Y (such as dictionaries and hash tables), and Y_{reads} denotes the extra memory space opened for genomic sequencing reads. To evaluate compression peak memory $Y_{cpm} = Y_{res} + Y_{reads}$ of Y on dataset F, PMFFRC performs the following steps:

Step 1: Randomly select x_1 and x_2 sets of sequencing data from F_i to construct pre-compression fastq format files $X_1 = \{x_0^1, x_1^1, x_2^1,\ldots, x_{v\times x_1-1}^1\}$ and $X_2 = \{x_0^2, x_1^2, x_2^2,\ldots, x_{v\times x_2-1}^2\}$, each group x_j^e contains description information, sequencing reads, and quality scores, where $e = 1,2$, $j = 0,1,2,\ldots,v\times x_e\text{-}1$, $i = 0,1,2,\ldots,v\text{-}1$ and $x_2 >> x_1$.

Step 2: Run compressor Y for pre-compressing datasets X_1 and X_2, getting Y's peak memory Y_{peak}^1 and Y_{peak}^2 on datasets X_1 and X_2.

Step 3: Estimate the compression peak memory Y_{cpm} of algorithm Y on datasets $F = \{F_0, F_1, \ldots F_{v\text{-}1}\}$ according to the formula (7):

$$Y_{cpm} = Y_{res} + Y_{reads} = Y_{peak}^1 + \frac{\left|Y_{peak}^2 - Y_{peak}^1\right|}{v \times (x_2 - x_1)} \times \sum_{i=0}^{v} \frac{|F_i|}{4} \quad (7)$$

In formula (7), $|F_i|/4$ denotes the total number of reads in F_i, where $i = 0,1,2,\ldots,v\text{-}1$.

Feature extraction

In pre-compression stage, PMFFRC estimated the maximum memory consumption Y_{cpm} of compressor Y on F. However, the calculated Y_{cpm} might exceed the system memory. Thus, PMFFRC compresses similar fastq files in batches via redundant reads clustering, using reads' feature vectors and user-preset safe memory threshold. However, converting string reads to digitized feature vectors is time-consuming. Therefore, PMFFRC utilizes CPU multi-cores to accelerate this stage in parallel.

Let Pr denotes the number of utilized CPU cores, $\bar{R}_i = [\dot{R}_0^i, \dot{R}_1^i, \dot{R}_2^i, \ldots, \dot{R}_{(|F_i|/4)-1}^i]$ denotes a numeral feature vector of R_i, and $\dot{R}_j^i$ is the feature value of read R_j^i. Referring to the design idea of the reads scoring model in FastqCLS (Lee and Song, 2022), PMFFRC employs Pr CPU cores to extract sequencing reads feature values in parallel through the data cycle division strategy (Cheng *et al.*, 2014), as shown in the formula (8):

$$\dot{R}_j^{i,p} = \left\lfloor \prod_{e=0}^{3} \left(\sum_{r=0}^{\left|R_j^{i,p}\right|} I\left(R_j^{i,p}[r] = E[e]\right) + 1 \right) / n \right\rfloor \quad (8)$$

In formula (8), $E = \{A, C, G, T\}$, $I\left(R_j^{i,p}[r] = E[e]\right)$ is an indicator function, p denotes the p-th CPU core, and $p = i \% Pr$, where $j = 0,1,2,\ldots,(|F_i|/4)\text{-}1$, $i = 0,1,2,\ldots,v\text{-}1$.

Similarity calculation

After obtaining the feature vector $\bar{R}_i = [\dot{R}_0^i, \dot{R}_1^i, \dot{R}_2^i, \ldots, \dot{R}_{(|F_i|/4)-1}^i]$ of R_i in F_i, PMFFRC evaluates the redundant reads similarity between different fastq files. Let S denotes a similarity collection of feature vectors $\bar{R}_a$ and $\bar{R}_b$. In practical scenarios, the data scale between F_a and F_b is unbalanced, usually. To offset the impact of fastq file size differences on similarity calculation between F_a and F_b, PMFFRC introduces correction parameters α and dice coefficient (Cha, 2007), uses Pr CPU cores, and improves the parallel similarity calculation model as shown in formula (9):

$$sim(\bar{R}_a^p, \bar{R}_b^p) = \alpha \times \frac{2 \times |\bar{R}_a^p \cap \bar{R}_b^p|}{|\bar{R}_a^p| + |\bar{R}_b^p|} \quad (9)$$

In formula (9), $sim(\bar{R}_a, \bar{R}_b)$ denotes the similarity value between the feature vectors $\bar{R}_a$ and $\bar{R}_b$. Where $\alpha = 1/(1 - \frac{||\bar{R}_a^p| - |\bar{R}_b^p||}{|\bar{R}_a^p| + |\bar{R}_b^p|})$, $p = s \,\%\, Pr$, $s = 0,1,2,\ldots,|S|\text{-}1$, $|S| = \frac{v \times (v-1)}{2}$, $a = 1,2,3,\ldots,v\text{-}1$, $b = 0,1,2,\ldots,v\text{-}2$, $a > b$.

Fastq Files Clustering

For computing friendly, PMFFRC converts the string reads files into numerical vectors to calculate S via formula (9). After that, it sorts S in descending order using the quicksort algorithm (Hoare, 1962) and performs fastq-files clustering. To select an appropriate parameter K, PMFFRC utilizes a two-level clustering parameter selection strategy. Specifically, PMFFRC first determines the files-level parameter K_1 and then slightly adjusts it to get the reads-level parameter K_2.

In actual experimental observations, the peak memory Y_{cpm} of Y shows a nonlinear growth trend with the fastq data scale growth of F in some cases (such as FastqCLS). Considering algorithm robustness, by modeling U_{ram} and Y_{cpm}, PMFFRC introduces an empirical correction factor β to artificial-fixed the files-level clustering parameter K_1, the calculation model as shown in formula (10):

$$K_1 = \left\lceil \beta \times \frac{Y_{reads}}{U_{ram} - Y_{res}} \right\rceil \quad (10)$$

In formula (10), the files-level clustering parameter K_1 was obtained under the safe memory threshold U_{ram} (less than system memory). However, if PMFFRC uses K_1 for fastq files clustering, which might lead to multiple reads in a cluster file. In that case, using PMMFRC for Y to compress clustered files might exceed the threshold U_{ram}. Therefore, PMFFRC dynamically and slightly adjusts the files-level K_1 to get the reads-level K_2 as the final fastq files clustering parameter K. Fig.1 illustrates the clustering of five sequencing fastq files using the proposed two-level (where $v = 5$ and $K_1 = 2$) clustering parameter selection strategy.

In Fig.1, $K_1 = 2$, and $M = 1060$ denotes the overall reads number. According to K_1 and M, PMFFRC gets the average number of reads in each cluster as $ave = \lfloor M/K_1 \rfloor = 530$. Fig.1(a) shows the similarity matrix calculated from formula (3). In Fig.1(b), PMFFR first detects that max(S) is $sim(\bar{R}_3, \bar{R}_0) = 0.92$, which indicates the redundant reads between fastq files F_3 and F_0 have the highest similarity, so it adds F_0 and F_3 to the cluster C_0. Then, PMFFRC calculates the total number of reads $|C_0|$ is 340 in C_0, which is less than ave, so it tries to add the fastq file F_1 in C_0. However, $|C_0|$ is 650 at this time, which is greater than ave, so it discards F_1 and obtains the first cluster $C_0 = \{F_0, F_3\}$. After the above steps, the first cluster C_0 has been received. Thus, PMFFRC

removes the elements in collection C_0 from the similarity matrix and selects the second cluster files. In Fig.1(c~d), after the first cluster has been bulit, the remaining components are $sim(\bar{R}_2, \bar{R}_1)$, $sim(\bar{R}_4, \bar{R}_1)$, $sim(\bar{R}_4, \bar{R}_2)$, and $sim(\bar{R}_4, \bar{R}_3)$, where $sim(\bar{R}_4, \bar{R}_1) = 0.90$ is the highest similarity score. Therefore, PMFFRC adds F_4 and F_1 to a new cluster $C_1 = \{F_1, F_4\}$. Now, $|C_1| = 460$, which is less than *ave*. If PMFFRC adds F_2 to cluster C_1, $|C_1|$ will exceed *ave*. Consequently, PMFFRC ends the second clustering stage to obtain cluster $C_1 = \{F_1, F_4\}$ and adds fastq file F_2 to another cluster $C_2 = \{F_2\}$.

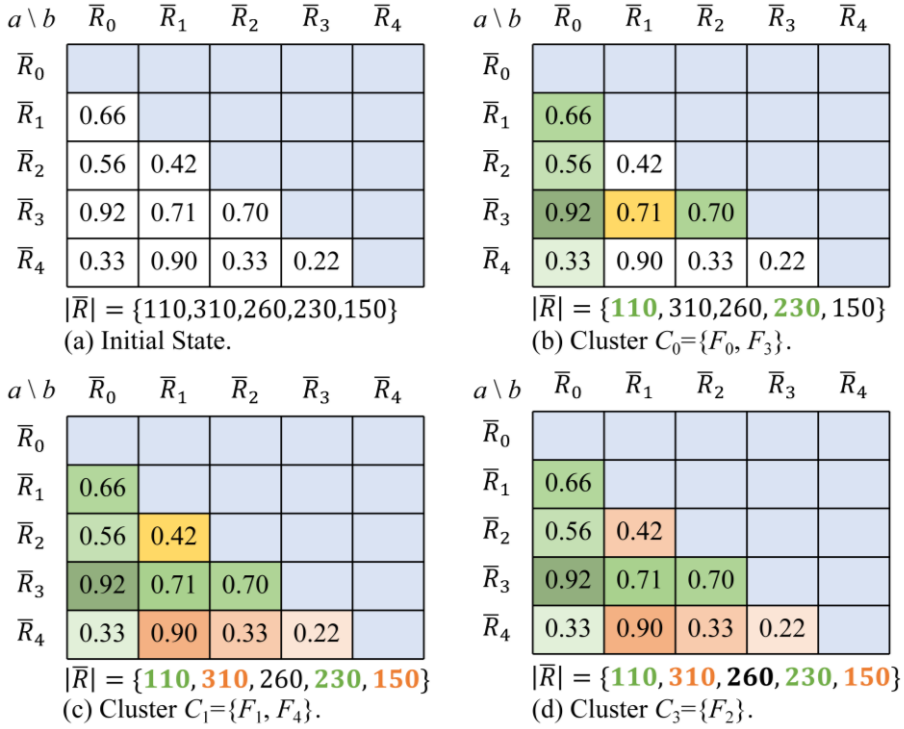


Fig.1 Example of clustering fastq files using a two-level parameter selection strategy. Via user-preset safe memory threshold and reads feature vectors, PMFFRC clusters similar fastq files together. Thus, the high-similarity files in the same cluster generate more highly similar redundant reads, which is more helpful for subsequent cascaded compressors.

Algorithm description and analysis

Algorithm 1 summarizes the proposed optimization method PMFFRC.

Algorithm 1: PMFFRC (Parallel Multi-Fastq-File Reads Clustering)

Input : $F = \{F_0, F_1, F_2, \ldots, F_{v-1}\}$, Y, U_{cpm}, Pr, x_1, x_2, β.

Output : Cluster record files C_k.info, where $k = 0,1,2,\ldots,K-1$.

Begin.

1 Employ parameters v, x_1, and x_2 to construct files X_1.fastq and X_2.fastq;

2 Use Y , X_1.fastq and X_2.fastq get Y_{peak}^1 and Y_{peak}^2;

3 Utilize β, Y_{peak}^1, and Y_{peak}^2 to calculate parameter K_1 via formulas (1) and (4);

4 Let $\bar{R} \leftarrow \{\varnothing\}$, $S \leftarrow \{\varnothing\}$, and $K_2 \leftarrow 0$;
5 Use *Pr* CPU cores for feature extraction on *F* via formula (2) to build $\bar{R}$;
6 Employ *Pr* CPU cores for similarity calculation using formula (3), record to *S*;
7 Use the quick sort algorithm to sort *S* in descending, obtain collection *RS*;
8 Get $M \leftarrow \lfloor \sum_{i=0}^{v} |\bar{R}_i| / K_1 \rfloor$;
9 $k \leftarrow 0$; // Dynamically determine the reads-level parameter $K_2 \leftarrow k + 1$.
10 Let $C_k \leftarrow \{\varnothing\}$, $N_k \leftarrow 0$, $M_k \leftarrow 0$, and $flag \leftarrow 0$;
11 **while** |*RS*| != 0 **do** // Dynamic clustering.
12 Add F_a and F_b recorded in RS_0 to cluster C_k, and remove RS_0 from *RS*;
13 $M_k \leftarrow |\bar{R}_a| + |\bar{R}_b|$, $N_k \leftarrow 2$;
14 **for** $i = 0$ **to** |*RS*| **do**
15 **if** $flag = 1$ **or** $M_k \geq M$ **then**
16 Write N_k ,C_k, and M_k to cluster record file C_k.info;
17 Remove the fastq files in C_k from *RS*;
18 Let $C_k \leftarrow \{\varnothing\}$, $N_k \leftarrow 0$, $M_k \leftarrow 0$, and $flag \leftarrow 0$;
19 $K_2 = k+1$, k++, break; // Update the clustering parameter K_2.
20 **end if** (15)
21 Extract F_a and F_b recorded by RS_i;
22 **if** $F_a \in C_k$ **and** $M_k + |\bar{R}_b| \geq M$ **then**
23 $flag \leftarrow 1$, continue;
24 **else if** $F_a \in C_k$ **and** $M_k + |\bar{R}_b| < M$ **then**
25 Add F_b to C_k. // Update the current cluster C_k.
26 $M_k \leftarrow M_k + |\bar{R}_b|$, N_k++;
27 **else if** $F_b \in C_k$ **and** $M_k + |\bar{R}_a| \geq M$ **then**
28 $flag = 1$, continue;
29 **else if** $F_b \in C_k$ **and** $M_k + |\bar{R}_a| < M$ **then**
30 Add F_a to C_k. // Update the current cluster C_k;
31 $M_k \leftarrow M_k + |\bar{R}_a|$, N_k++;
32 **else** continue;
33 **end if** (22)
34 **end for** (14)
35 **end while** (11)
End.

Let $m = \frac{(\sum_{i=1}^{v} |F_i|)/4}{v}$, in Algorithm 1, the worst time to construct pre-compression fastq format files X_1.fastq and X_2.fastq in step 1 is O(4×n×v×(x_1+x_2)). To pre-compress X_1.fastq and X_2.fastq in step 2 is O(y). To calculate the files-level clustering parameter K_1 in step 3 is O(m×v). Step 4 consumes O(1) time. The worst time for parallel feature

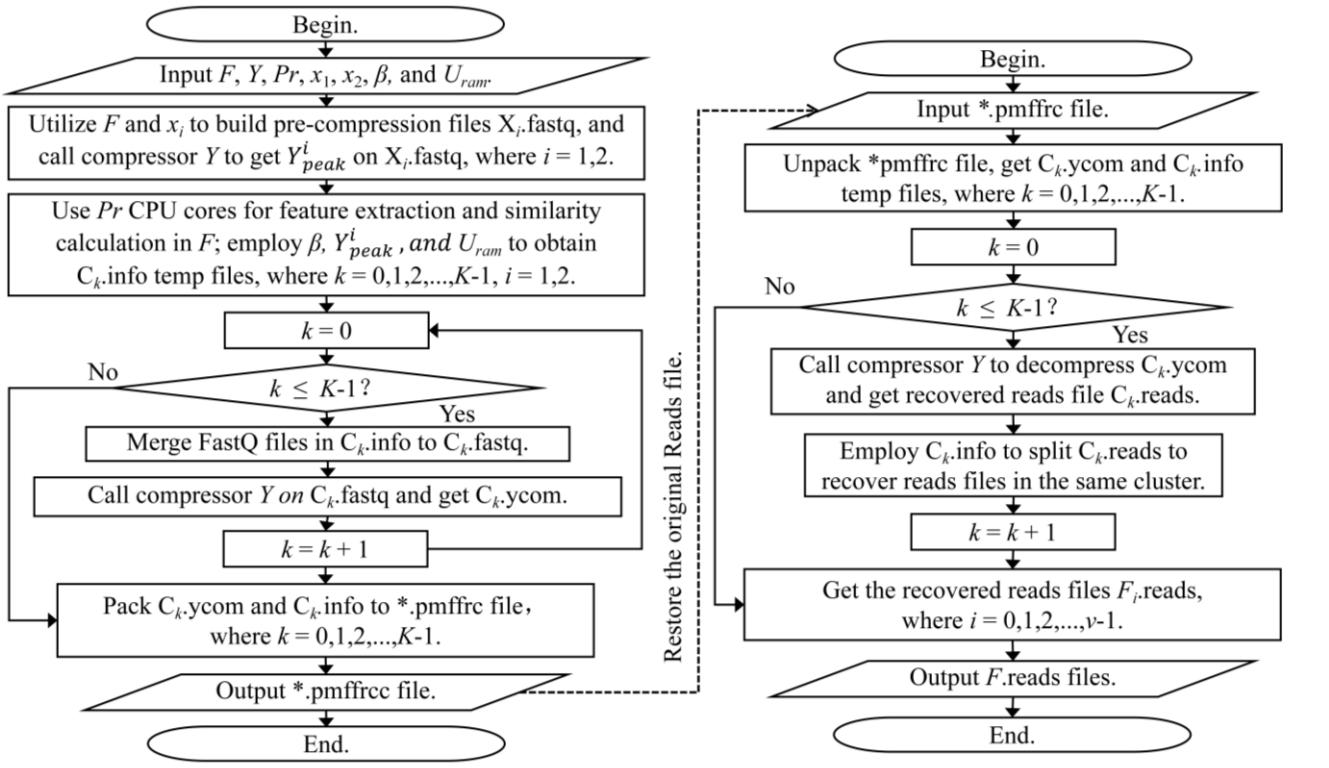
extraction in step 5 is O($\frac{m\times v\times n}{Pr}$). To parallel calculate the files similarity in step 6 is O($\frac{v\times(v-1)\times m}{2\times Pr}$) = O($\frac{m\times v^2}{Pr}$). Sortig S in descending order in step 7 is O($v^2 \times log_2^{v^2}$). Step 8 time consumption is O(1). The worst time for the inner *for*-loop in steps 14~34 is O(v^2). The worst time for the outer *while*-loop in steps 12~35 is O($K\times v^2$), where $K = K_2$. Due to parameters x_1, x_2 and K are all constants and $m\times v>>y$, $m>>v$. Thus, the time complexity of PMFFRC is O(max{$4\times n\times v\times(x_1+x_2)$, y , $\frac{m\times v\times n}{Pr}$, $\frac{m\times v^2}{Pr}$, $v^2\times log_2^{v^2}$, $K\times v^2$}) = O(max{$\frac{m\times v\times n}{Pr}$, $v^2 \times log_2^{v^2}$}) = O($\frac{m\times v\times n}{Pr}$).

In Algorithm 1, the space required to calculate the compression peak memory Y_{cpm} on F is O(y). The memory for parallel feature extraction and similarity computation is O($Pr\times m$). The space required for set $\bar{R}$ is O($m\times v$). The space required for S and RS is O($\frac{v\times(v-1)}{2}$) = O(v^2). The worst space usage of C_k is O(v), where k = 0,1,2,…,K-1. Thus, the space complexity of the proposed algorithm PMFFRC is O(max{y, $Pr\times m$, $m\times v$, v^2}) = O($m\times(v+Pr)$), where $m\times v>>y$ and $m>>v$.

S3. Joint compression and decompression

When obtaining the cluster record files, C_k.info, of the sequencing files collection F = {F_0, F_1,...F_{v-1}}, the fastq files in each cluster C_k are merged according to temp files C_k.info, where k = 0,1, 2,…, K-1. Therefore, dedicated state-of-the-art compressors can be used to compress these clustered files. Fig.2 shows the overall processing workflow of the PMFFRC optimizes algorithm Y to compress and decompress fastq format files collection F = {F_0, F_1,...F_{v-1}}.

In Fig.2(a), C_k.ycom denote the compressed reads files by compressor Y, and the *.pmffrc file is the optimized compressed files, where k = 0,1,2,…,K-1. In Fig.2(b), F_i.reads denote the recovered decompressed files corresponding to F_i.fastq, where i = 0,1,2,…,v-1.



(a) Optimizing Compression Workflow for Y via PMFFRC. (b) Joint Compression Restores Raw Sequencing Reads Data.

Fig.2 The overall processing workflow of the optimized cascaded compressor Y via PMFFRC, on fastq format sequencing files collection F.

S4. Experimental configuration and datasets

Evaluation experiments were carried out on a Sugon-7000A supercomputer system of the National High-Performance Computing Center Nanning Branch (https://hpc.gxu.edu.cn). Computing nodes are equipped with 2*Intel Xeon Gold 6230 CPU (2.1 Ghz, 40 cores), 2*NVIDIA Tesla-T4 GPU (16 GB DRAM, 2560 cores), 512 GB DDR4 SDRAM, and 8*900 GB external storage. The PMFFRC algorithm for optimizing large-scale reads compression was implemented by C++11 and OpenMP. The overall workflow has been encapsulated into the PMFFRC toolkit. Section S5 gives the installation and configuration details of the PMFFRC toolkit. Three large-scale datasets from the NCBI (https://www.ncbi.nlm.nih.gov) database were used to evaluate optimization effects. Table S1 gives the details of the experimental datasets. For complete NCBI registration numbers, see Section S6.

Table S1. Datasets used for optimizing large-scale genomic sequencing reads compression.

dataset name (species)	sequencing platform (method)	number of reads	total size (KiloByte)	reads size (KiloByte)	length (bp)	number of files
Homo sapiens	NextSeq-550 (SE)	507400829	116656482	38562464	75	24
Cicer arietinum	HiSeq-2000 (PE)	2060956042	713658327	187211471	90	60
Salvelinus fontinalis	Ion-Torrent (SE)	757458105	198877977	60968323	80	360
total	---	3325818976	981.515GB	273.459GB	---	424

Notes: Datasets download by employing sra-tools (https://github.com/ncbi/sra-tools). GB: Gigabyte.

S5. Installation and Configuration

Currently, the PMFFRC optimization algorithm only supports compressors HARC (Chandak *et al.*, 2018), SPRING (Chandak *et al.*, 2019), FastqCLS (Lee and Song, 2022), and Mstcom (Liu and Li, 2021). Since PMFFRC has high versatility, it can be adapted to more reads compressors by making appropriate modifications in script files. The specific usage workflow is as follows:

1. Clone PMFFRC toolkit from GitHub:

   ```
   git clone https://github.com/fahaihi/PMFFRC.git
   ```

2. Turn to the PMMFRC directory：

   ```
   cd PMFFRC
   ```

3. Compile PMFFRC:

   ```
   ./install.sh
   ```

4. Turn to `/src` directory, and configure compressors script files `*_compressor.sh` and `*_decompressor.sh`. Installing and configuring HARC (2018), SPRING (2019), Mstcom (2021), and FastqCLS (2022) compression algorithms, see the following GitHub repositories:

   ```
   HARC(2018)：https://github.com/shubhamchandak94/HARC
   SPRING(2019)：https://github.com/shubhamchandak94/SPRING
   Mstcom(2021)：https://github.com/yuansliu/Mstcom
   FastqCLS(2022)：https://github.com/Krlucete/FastqCLS
   ```

5. After configuring `src/*_compressor.sh` and `src/*_decompressor.sh` compressors scripts files, run `./PMFFRC` with the following Linux command:
   Compression -> Compress Multi-Fastq Files:

   ```
   ./PMFFRC [-c multi-fastq-files path]
            [-y cascading algorithm used. such as harc]
            [-t num of threads. --Default = 20]
            [-u user-defined Uram size. --Default 10 GB]
            [-q write quality values and read ids to .quality && .id files]
            [-e clean temp files. --Default "false"]
   ```

   DECompression -> DECompress Multi-Fastq Files:

   ```
   ./PMFFRC [-d decompression *.pmffrc format file]
            [-t num of CPU cores. --Default = 20]
            [-e clean temp files. --Default = "false"]
   ```

   Help -> Print Help Message:

```
./PMFFRC -h
```

Here are some examples of HARC using the algorithm PMMFRC to optimize large-scale reads compression.

1. Compress multiple fastq files in /userdir/data/testdir directory using 20 CPU cores and 40 GB of secure memory:

   ```
   ./PMFFRC -c /userdir/data/testdir -y harc -t 20 -u 40 -q -e
   ```

2. If you don't want to save *.quality, *.id, and * temp files, use this command:

   ```
   ./PMFFRC -c /userdir/data/testdir -y harc -t 20 -u 40
   ```

3. Decompress the /userdir/data/testdir.pmffrc file using 20 CPU cores while keeping the intermediate result files:

   ```
   ./PMFFRC -d /userdir/data/testdir.pmffrc -y harc -t 20
   ```

4. Print help information:

   ```
   ./PMFFRC -h
   ```

Our experimental time and memory consumption tests using the following command:

```
/bin/time -v -p [test script]
```

Notes: In our experiments, we set PMFFRC to select the first x_1 and x_2 groups of fastq format sequencing data for maximum compression memory evaluation. The parameters x_1, x_2, and β are configured in the `src/*_compressor.sh` script files. We recommend $x_1 = 100$ and $x_2 = 100100$ as the basic settings for the memory evaluation stage. According to our experience, we recommend $\beta_{\text{HARC}} = 1.05$, $\beta_{\text{SPRING}} = 0.30$, $\beta_{\text{FastqCLS}} = 0.28$, and $\beta_{\text{Mstcom}} = 0.75$ as the artificial-fixed empirical correction factor for cascaded compressors HARC, SPRING, Mstcom, and FastqCLS.

S6. Datasets acquisition

Three large-scale datasets, Homo sapiens, Salvelinus fontinalis, and Cicer arietinum, from NCBI Sequence Read Archive (https://www.ncbi.nlm. nih.gov/sra) were used for experimental testing.

1. For Homo sapiens dataset, we randomly selected the following registration numbers:

   ```
   ERR7091240-ERR7091243; ERR7091245-ERR7091248; ERR7091253-
   ERR7091256; ERR7091258-ERR7091269 (24 SE-Files).
   ```

2. For Cicer arietinum dataset, we randomly selected the following registration numbers:

   ```
   SRR13556190-SRR13556217; SRR13556220; SRR13556224 (60 PE-Files).
   ```

3. For Salvelinus fontinalis dataset, we randomly selected the following registration numbers:

   ```
   SRR11994925-11995284 (360 SE-Files).
   ```

The script files implementation for downloading these datasets are given in the PMFFRC algorithm command line open source tool (https://github.com/fahaihi/PMFFRC). The specific usage method is as follows:

```
cd PMFFRC/data
nohup ./NextSeq-550_Homo_sapiens_SE.sh &
nohup ./HiSeq-2000_Cicer_arietinum_PE.sh &
nohup ./Ion-Torrent_Salvelinus_fontinalis_SE.sh &
```

S7. Speedup and relative memory consumption

The PMFFRC algorithm converts string sequencing reads into numerical feature vectors to simplify subsequent clustering calculations. However, the number of reads in fastq sequencing files usually reaches millions or even billions, which brings significant challenges for sequencing reads feature extraction and fastq-files clustering. Therefore, the PMFFRC algorithm utilizes CPU multi-cores to accelerate these computationally intensive steps to improve the algorithm's running efficiency. Tables S1~S3 show the clustering time (*Time*) and memory consumption (*Mem*) of the compressors HARC, SPRING, Mstcom, and FastQCLS employing the PMFFRC algorithm at different CPU cores (*Pr*).

References

Cha,S.-H. (2007) Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. *International Journal of Mathematical models and Methods in Applied Sciences*, **1**, 300–307.

Chandak,S. *et al.* (2018) HARC： Compression of genomic sequencing reads via hash-based reordering: algorithm and analysis. *Bioinformatics*, **34**, 558–567.

Chandak,S. *et al.* (2019) SPRING: a next-generation compressor for FASTQ data. *Bioinformatics*, **35**, 2674–2676.

Cheng,J. *et al.* (2014) Professional CUDA c programming John Wiley & Sons.

Hoare,C.A.R. (1962) Quicksort. *The Computer Journal*, **5**, 10–16.

Lee,D. and Song,G. (2022) FastqCLS: a FASTQ compressor for long-read sequencing via read reordering using a novel scoring model. *Bioinformatics*, **38**, 351–356.

Liu,Y. and Li,J. (2021) Hamming-shifting graph of genomic short reads: Efficient construction and its application for compression. *PLOS Computational Biology*, **17**, e1009229.

Tang,T. and Li,J. (2021) Transformation of FASTA files into feature vectors for unsupervised compression of short reads databases. *Journal of bioinformatics and computational biology*, **19**, 2050048.

Table S2. Clustering time and memory consumption of PMFFRC on the Homo sapiens dataset using different CPU cores.

Algorithm / Cores.*num*		*Pr* = 1	*Pr* = 2	*Pr* = 4	*Pr* = 6	*Pr* = 8	*Pr* = 10	*Pr* = 12	*Pr* = 14	*Pr* = 16	*Pr* = 18	*Pr* = 20
HARC	*Time*	00:14:22	00:10:09	00:07:08	00:06:19	00:04:58	00:03:58	00:03:57	00:03:26	00:03:12	**00:02:43**	00:02:59
	Speedup	1.000	1.415	2.014	2.274	3.622	3.622	3.637	4.185	4.490	**5.288**	4.816
	Mem	**2372648**	2614520	2933904	3076132	3972188	3972188	3971680	4246744	4483728	4666676	4812400
	R-Mem	**1.000**	1.102	1.237	1.296	1.674	1.674	1.674	1.790	1.890	1.997	2.028
SPRING	*Time*	00:14:16	00:10:21	00:07:03	00:06:19	00:04:29	00:04:08	00:04:02	00:03:25	00:03:21	**00:02:47**	00:03:16
	Speedup	1.000	1.378	2.024	2.259	3.182	3.452	3.537	4.176	4.259	**5.126**	4.435
	Mem	**2362340**	2588024	2841932	3009032	3545828	3829884	3873012	4317440	4485756	4705356	4838360
	R-Mem	**1.000**	1.096	1.203	1.274	1.501	1.621	1.639	1.828	1.899	1.992	2.048
Mstcom	*Time*	00:23:21	00:15:23	00:10:44	00:10:31	00:08:07	00:05:08	00:05:02	00:05:31	00:04:44	00:04:35	**00:03:55**
	Speedup	1.000	1.518	2.175	2.220	2.877	4.549	4.639	4.233	4.933	5.095	**5.962**
	Mem	**2373200**	2976864	2819972	3031476	3390920	3802428	3962944	4151928	4456832	4727912	4942600
	R-Mem	**1.000**	1.254	1.188	1.277	1.429	1.602	1.670	1.750	1.878	1.992	2.093
FastqCLS	*Time*	00:30:33	00:19:37	00:12:10	00:11:22	00:10:08	00:09:59	00:07:20	00:06:35	00:08:36	**00:06:37**	00:07:33
	Speedup	1.000	1.557	2.511	2.688	3.015	3.060	4.166	4.641	3.552	**4.617**	4.046
	Mem	3207952	**2533636**	3121756	3436880	3557980	3846956	3882096	4194640	4492116	4669120	4792608
	R-Mem	1.000	**0.790**	0.973	1.072	1.109	1.199	1.210	1.308	1.400	1.455	1.494

Notes: Compression parameters: U_{ram} = 30GB, T = 8, β_{HARC} = 1.05, β_{SPRING} = 0.30, β_{Mstcom} = 0.75, β_{FastqCLS} = 0.28, x_1 = 100, x_2 = 100100. *Speedup* = $Time_{Pr=1}/Time_{Pr=i}$, *R-Mem* = $Mem_{Pr=i}/Mem_{Pr=1}$, wherer i = 1,2,4,6,8,10,12,14,16,18,20。The best results in the table are shown in boldface.

Table S2 shows that on the Homo sapiens dataset, the clustering *Speedup* of PMFFRC algorithm cascade HARC, SPING, Mstcom, and FastqCLS algorithms gradually increases with the increase of CPU cores and reaches the peak at 18~22 cores. In clustering peak memory *Men*, the memory overhead increases with increasing the number of parallel cores *Pr*.

Table S3. Clustering time and memory consumption of PMFFRC on the Salvelinus fontinalis dataset using different CPU cores.

Algorithm / Cores.*num*		*Pr* = 1	*Pr* = 2	*Pr* = 4	*Pr* = 6	*Pr* = 8	*Pr* = 10	*Pr* = 12	*Pr* = 14	*Pr* = 16	*Pr* = 18	*Pr* = 20
HARC	*Time*	02:04:54	01:39:43	01:02:52	00:43:21	00:43:19	00:30:09	00:30:20	00:24:27	00:24:53	00:21:47	**00:21:44**
	Speedup	1.000	1.253	1.987	2.881	2.883	4.143	4.118	5.108	5.019	5.734	**5.747**
	Mem	**3458472**	3551188	3654236	3765168	3823460	3991220	4000684	4171728	4111012	4327564	4285256
	R-Mem	**1.000**	1.027	1.057	1.087	1.106	1.154	1.157	1.206	1.189	1.251	1.239
SPRING	*Time*	02:08:19	01:39:16	1:02:12	00:42:50	00:41:43	00:29:23	00:30:15	00:23:25	00:28:24	**00:20:31**	00:23:57
	Speedup	1.000	1.293	2.063	2.996	3.076	4.367	4.242	5.480	4.518	**6.254**	5.358
	Mem	**3539540**	3555456	3655992	3850668	3801796	4005040	3950588	4245532	4065220	4363096	4268504
	R-Mem	**1.000**	1.004	1.033	1.088	1.074	1.132	1.116	1.199	1.149	1.233	1.206
Mstcom	*Time*	03:58:37	02:41:37	01:45:42	01:11:48	01:08:53	00:55:11	00:44:42	00:38:51	00:39:31	**00:35:19**	00:38:01
	Speedup	1.000	1.476	3.688	3.323	3.464	4.324	5.338	6.142	6.038	**6.756**	4.969
	Mem	**11588048**	11736724	11649436	11665712	11703908	11532164	11590444	11631524	11611521	11739684	11735280
	R-Mem	**1.000**	1.013	1.005	1.007	1.010	0.995	1.000	1.004	1.002	1.013	1.013
FastqCLS	*Time*	06:31:21	04:11:57	02:25:06	01:49:01	01:38:14	01:22:16	00:57:34	00:49:16	00:50:02	00:45:48	**00:42:47**
	Speedup	1.000	1.553	2.697	3.560	3.984	5.341	6.798	7.944	6.519	8.545	**9.147**
	Mem	20438244	**20104044**	20810742	2164091	21517440	21479524	21441608	22757064	21506268	22643344	23780420
	R-Mem	1.000	**0.984**	1.018	1.036	1.053	1.051	1.049	1.113	1.052	1.108	1.164

Notes: Compression parameters: U_{ram} = 30GB, T = 8, β_{HARC} = 1.05, β_{SPRING} = 0.30, β_{Mstcom} = 0.75, β_{FastqCLS} = 0.28, x_1 = 100, x_2 = 100100. *Speedup* = $Time_{Pr=1}/Time_{Pr=i}$, *R-Mem* = $Mem_{Pr=i}/Mem_{Pr=1}$, wherer i = 1,2,4,6,8,10,12,14,16,18,20。The best results in the table are shown in boldface.

Table S3 show that on the Salvelinus fontinalis dataset, the PMFFRC algorithm cascaded HARC, SPING, Mstcom, and FastqCLS algorithms reached the *speedup* peak when the *Pr* takes 20, 18, 18, and 20, respectively. The peak *speedup* (total) was higher than the Homo sapiens dataset, which shows that the parallel acceleration gains of the PMFFRC algorithm significantly increased by the size of the dataset and the number of files.

Table S4. Clustering time and memory consumption of PMFFRC on the Cicer arietinum dataset using different CPU cores.

Algorithm / Cores.*num*		*Pr* = 1	*Pr* = 2	*Pr* = 4	*Pr* = 6	*Pr* = 8	*Pr* = 10	*Pr* = 12	*Pr* = 14	*Pr* = 16	*Pr* = 18	*Pr* = 20
HARC	*Time*	2:10:33	1:31:39	00:56:39	00:38:57	00:31:30	00:27:16	00:23:30	00:20:12	00:17:03	00:16:20	**00:15:37**
	Speedup	1.000	1.424	2.305	3.352	4.144	4.788	5.555	6.463	7.657	7.993	**8.360**
	Mem	**8464544**	8783488	9387212	9976220	10612548	11309580	11870800	12542348	13040740	13646532	13917052
	R-Mem	**1.000**	1.038	1.109	1.179	1.254	1.336	1.402	1.482	1.541	1.612	1.644
SPRING	*Time*	02:10:01	01:32:02	00:56:37	00:39:12	00:31:20	00:26:48	00:22:35	00:19:46	00:16:53	00:16:16	**00:15:32**
	Speedup	1.000	1.413	2.296	3.317	4.149	4.851	5.757	6.578	7.701	7.993	8.370
	Mem	**8515380**	8767936	9406268	10034804	10583420	11270940	11815216	12470212	12986956	13628280	13931704
	R-Mem	**1.000**	1.030	1.105	1.178	1.243	1.324	1.388	1.464	1.525	1.600	1.636
Mstcom	*Time*	03:39:50	02:34:15	01:37:14	01:02:00	00:35:19	00:36:57	00:35:21	00:31:41	00:28:05	**00:28:03**	00:28:04
	Speedup	1.000	1.406	2.230	3.497	6.140	5.868	6.134	6.844	7.721	**7.730**	7.726
	Mem	**8432776**	8716100	9347320	9953996	10509312	11153612	11783056	12463852	12949340	13646328	14059428
	R-Mem	**1.000**	1.034	1.108	1.180	1.246	1.323	1.397	1.478	1.536	1.618	1.667
FastqCLS	*Time*	03:13:20	02:14:56	01:23:15	01:03:16	00:45:29	00:40:44	00:27:20	00:23:54	00:21:27	**00:21:01**	00:24:01
	Speedup	1.000	1.433	2.322	3.056	4.251	4.746	7.073	8.089	9.013	**9.199**	8.050
	Mem	**8517992**	8769584	9355028	9988872	10571216	11284636	11721968	12382248	13130388	13520448	14063956
	R-Mem	**1.000**	1.030	1.098	1.173	1.241	1.325	1.376	1.454	1.541	1.587	1.651

Notes: Compression parameters: U_{ram} = 30GB, T = 8, β_{HARC} = 1.05, β_{SPRING} = 0.30, β_{Mstcom} = 0.75, β_{FastqCLS} = 0.28, x_1 = 100, x_2 = 100100. *Speedup* = $Time_{Pr=1}/Time_{Pr=i}$, *R-Mem* = $Mem_{Pr=i}/Mem_{Pr=1}$, wherer i = 1,2,4,6,8,10,12,14,16,18,20。The best results in the table are shown in boldface.

Similar to the experimental results in Table S2 and Table S3, the four cascaded algorithms HARC, SPRING, Mstcom, and FastqCLS in Table S4 also reach the peak *speedup* when the *Pr* value takes 18~20. In terms of clustering memory consumption, the PMFFRC algorithm uses CPU multi-cores for parallel feature extraction and similarity calculation. Each CPU core needs to open additional memory to store intermediate calculation results temporarily, so the peak memory overhead in parallel mode is higher than in serial mode (*Pr* = 1).