Supplementary Material

Table of Contents

1 Preface 1

2 Datasets 1

3 Performance Benchmarking 2

4 SAM Fields/Columns 2

5 Evaluation Categories 3

6 Tools 3

6.1 Tsc 3

6.2 Quip 4

6.3 DeeZ 4

6.4 Scramble (CRAM 3.0) 5

7 References 5

1 Preface

The compression of aligned reads in the form of Sequence/Alignment Map (SAM) format files [1] is more favorable compared to the compression of unaligned reads usually provided as FASTQ files [2]. The redundancy introduced by a high sequencing coverage[1] cannot be exploited by the compression of unaligned reads. The process of alignment with tools such as BWA [3] or Bowtie2 [4], producing SAM files, renders the redundant information directly exploitable to compression tools.

This evaluation compares tools for the compression of aligned reads which are in the form of SAM files. Some of the tools used in the scope of this report also support BAM files as input. Nevertheless, since the long-term task is to replace the BAM format, we solely focus on SAM-to-X-to-SAM compression and decompression.

2 Datasets

We used already aligned data from the Moving Picture Experts Group (MPEG) genomic information compression database (GIDB) [5] to evaluate the performance of the chosen compression tools. Most of the data in the GIDB is provided in the form of BAM files. We first converted the data back to the SAM format using SAMtools. The option –h ensures that the SAM header is preserved during BAM-to-SAM conversion.

```
$ samtools view –h file.bam > file.sam
```

[1] Each genomic “column” is read out multiple times during the sequencing process; the average is referred to as “coverage”.

An overview about the dataset is given in Table 1. We assigned identifiers in column 1 for brevity. If not specified otherwise, the technology information in column 2 has been obtained from the SAM header[2]. The coverage has been computed with SAMtools and awk.

```
$ samtools depth file.bam | awk '{sum+=$3} END { print "Coverage: ",sum/NR}'
```

Table 1: Datasets from the GIDB used for evaluation

Identifier	Technology	Category	BAM size (GB)	Record count	Coverage
cancermixed	Illumina	Human	123.0	942,395,158	26x
DH10B	Illumina MiSeq	Bacterial	1.4	13,175,679	448x
LID8465	Illumina RNA-Seq	Human	13.0	246,476,391	16x
ERR317492WGS	Illumina HiSeq 2000	Human	6.1	56,463,236	2.3x
ERP002490	Illumina HiSeq 2000	Human	106.0	1,286,194,550	26x
NA21144	Illumina HiSeq 2000	Human	1.0	10,147,680	7.4x

The datasets ERR317492WGS and NA21144 have been used in the Scramble paper [6], too. Furthermore, the dataset DH10B has been used in the DeeZ paper [7].

3 Performance Benchmarking

All measurements have been performed on an Intel® Core™ i7-3770K CPU with 4 cores using hyper-threading @ 3.90 GHz and 32 GB RAM. Timing and memory measurements have been performed with GNU time using the following format string:

```
$ /usr/bin/time -f "real: %es\nuser: %Us\nsys: %Ss\nmax RSS: %MkB" command
```

%e gives the elapsed real (i.e. wall-clock) time in seconds, %U the CPU time in user space, %S the CPU time in kernel space, and %M yields the maximum resident set size of the supervised process in kB. The total elapsed CPU time can be computed with %U+%S.

4 SAM Fields/Columns

A SAM file is structured into the SAM header and an alignment section, made up of SAM records. Each SAM record occupies one line in the SAM file and is made up of 12 fields/columns which are separated by tab stops. The names to identify the columns in Table 2 have been chosen according to the SAM format specification [1]. To get the total SAM file size, we have to count the SAM header (HEAD), and control symbols (tab stops and line breaks), too.

Table 2: SAM fields/columns

Field/Column	Name	Remarks/Evaluation Category
1	QNAME	Ident
2	FLAG	Aux
3	RNAME	Nuc
4	POS	Nuc
5	MAPQ	Aux
6	CIGAR	Nuc
7	RNEXT	Paired
8	PNEXT	Paired
9	TLEN	Paired

[2] @RG line, PL tag

10	SEQ	Nuc
11	QUAL	Qual
12	OPT	Aux
--	*CRTL*	*Tab stops (\t) and line breaks (\n)*
--	*HEAD*	*SAM header*

5 Evaluation Categories

In order to compare different SAM compression tools, SAM fields be associated with categories as shown in Table 3. It is questionable where to place FLAG and MAPQ. However, most codecs consider it as being auxiliary data and so do we.

Table 3: Evaluation categories

Category	Associated SAM Fields	Remark
Aux	FLAG, MAPQ, OPT	Auxiliary data
Ident	QNAME	Read identifier, as also stored in FASTQ files
Nuc	RNAME, POS, CIGAR, SEQ	Nucleotide sequences with associated mapping information
Paired	RNEXT, PNEXT, TLEN	Information about paired-end reads
Qual	QUAL	Quality scores

6 Tools

Several tools have been selected for evaluation. They are listed in Table 4.

Table 4: Selected compression tools

File format	Tool	Version	Reference-based	Random Access	Remark
TSC 1.0	tsc	1.0	N	Y	
TSC 1.2		1.2	N	Y	
QP	quip	1.1.8	N	N	Does not preserve RNEXT and OPT
	quip -a		N	N	
DZ	deez	1.0	Y	Y	
		1.1	Y	Y	
CRAM 3.0	scramble	1.14.6	Y	Y	

6.1 Tsc

The tsc software is structured into codec categories that have been assigned to evaluation categories as shown in Table 5.

Table 5: Tsc codec categories

Codec Category	Input	Evaluation Category
Aux	FLAG, MAPQ, OPT	Aux
Ident	QNAME	Ident
Nuc	RNAME, POS, CIGAR, SEQ	Nuc
Paired	RNEXT, PNEXT, TLEN	Paired
Qual	QUAL	Qual

Tsc has been compiled with the flag `-O2` and is a single-threaded tool. The option `-s` enables printing detailed statistics after compression.

```
$ tsc -s file.sam -o file.sam.tsc
```

```
$ tsc –d -s file.sam.tsc –o file.sam.tsc.sam
```

6.2 Quip

The Quip [8] software is structured into codec categories that have been assigned to evaluation categories as shown in Table 6. In some cases, Quip does not preserve RNEXT correctly ("=" gets mixed with "*"). Furthermore, the column order in OPT is not preserved.

Table 6: Quip codec categories

Codec Category	Input	Evaluation Category
ID	QNAME	Ident
Aux	OPT	Aux
Seq	FLAG, RNAME, POS, MAPQ, CIGAR, RNEXT, PNEXT, TLEN, SEQ	Aux/Nuc/Paired
Qual	QUAL	Qual

The options `-lv` print detailed statistics for the compressed `.qp` file.

```
$ quip –lv file.sam.qp
```

Quip in the "normal" (non-reference-based) mode.

```
$ quip –i sam –c file.sam > file.sam.qp
$ quip –d –o sam file.sam.qp –c > file.sam.qp.sam
```

Quip in the de-novo assembly mode.

```
$ quip –i sam –a –c file.sam > file.sam.qp
$ quip –d –o sam -c file.sam.qp > file.sam.qp.sam
```

6.3 DeeZ

The DeeZ [7] software is structured into codec categories that have been assigned to evaluation categories as shown in Table 7.

Table 7: DeeZ codec categories

Codec Category	Input	Evaluation Category
sequence	POS, CIGAR, SEQ, RNAME	Nuc
editOp	POS, CIGAR, SEQ	Nuc
readName	QNAME	Ident
mapFlag	FLAG	Aux
mapQual	MAPQ	Aux
quality	QUAL	Qual
pairedEnd	RNEXT, PNEXT, TLEN	Paired
optField	OPT	Aux

First, we have to build an index for the reference FASTA file using SAMtools. Alternatively, DeeZ can build its own index file.

```
$ samtools faidx ref.fa
```

The option `-r` indicates the reference to be used. Option `–S` enables the printing of detailed compression statistics.

```
$ deez –S –r ref.fa file.sam –o file.sam.dz
$ deez –d –r ref.fa file.sam.dz –o file.sam.dz.sam
```

6.4 Scramble (CRAM 3.0)

The codec categories for the CRAM 3.0 format are shown in Table 8.

First, we have to build an index for the reference FASTA file using SAMtools.

```
$ samtools faidx ref.fa
```

By default, Scramble uses four threads and the default compression level is '-5'. We configure Scramble to output the CRAM 3.0 format.

```
$ scramble –I sam –O cram -5 –V 3.0 file.sam file.sam.cram
$ scramble –I cram –O sam -5 –V 3.0 file.sam.cram file.sam.cram.sam
```

Table 8: CRAM codec categories

Codec Category	Evaluation Category
RN	Ident
QS	Qual
IN	Nuc
SC	Nuc
BF	Aux
CF	Paired
AP	Nuc
MQ	Aux
NS	Paired
MF	Paired
TS	Paired
NP	Paired
NF	Paired
FN	Nuc
FC	Nuc
FP	Nuc
DL	Nuc
BA	Nuc
BS	Nuc
TL	Aux
RI	Nuc
Three-letter ones	Aux

7 References

[1] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and 1000 Genome Project Data Processing Subgroup, "The Sequence Alignment/Map format and SAMtools," *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[2] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants," *Nucleic Acids Res.*, vol. 38, no. 6, pp. 1767–1771, 2010.

[3] H. Li and R. Durbin, “Fast and accurate short read alignment with Burrows-Wheeler transform.,” *Bioinformatics*, vol. 25, no. 14, pp. 1754–60, 2009.

[4] B. Langmead and S. L. Salzberg, “Fast gapped-read alignment with Bowtie 2.,” *Nat. Methods*, vol. 9, no. 4, pp. 357–9, 2012.

[5] C. Alberti, M. Mattavelli, L. Chiariglione, I. Xenarios, N. Guex, H. Stockinger, T. Schuepbach, P. Kahlem, C. Iseli, D. Zerzion, D. Kuznetsov, Y. Thoma, E. Petraglio, C. Sahinalp, I. Numanagic, and J. Delgado, “Database for Evaluation of Genome Compression and Storage,” Geneva, 2015.

[6] J. K. Bonfield, “The Scramble conversion tool.,” *Bioinformatics*, vol. 30, no. 19, pp. 2818–9, Oct. 2014.

[7] F. Hach, I. Numanagić, and S. C. Sahinalp, “DeeZ: reference-based compression by local assembly.,” *Nat. Methods*, vol. 11, no. 11, pp. 1082–4, Nov. 2014.

[8] D. C. Jones, W. L. Ruzzo, X. Peng, and M. G. Katze, “Compression of next-generation sequencing reads aided by highly efficient de novo assembly,” *Nucleic Acids Res.*, vol. 40, no. 22, pp. e171–e171, 2012.