Next Generation Sequencing Data and its Compression

To cite this article: Bruno Carpentieri 2019 IOP Conf. Ser.: Earth Environ. Sci. 362 012059

View the article online for updates and enhancements.
Next Generation Sequencing Data and its Compression

Bruno Carpentieri

Dipartimento di Informatica, Università di Salerno, Via Giovanni Paolo II, 132, 84084 Fisciano (SA), Italy
bcarpentieri@unisa.it

Abstract. Over the past few years the amount of digital memory and network traffic used by sequenced biological data has increased dramatically. Genomic projects such as HapMap, 1000 Genomes, etc., have come to the collection and description of genomes of 2,504 individuals from 26 populations, and they contributed to exponential growth of databases of this type and to the development of increasingly efficient technologies. Thanks to the large-scale sequencing of samples of DNA, the interest and the new research in these areas by the scientific community are suddenly grown. In a very short time researchers have developed hardware tools, analysis software, algorithms, private databases and infrastructures to support genomics. In this paper we analyse different approaches for compressing digital files generated by Next-Generation Sequencing tools containing nucleotide sequences and we discuss and evaluate the compression performance of generic compression tools such as gzip and bzip2 by confronting them with a specific system that was designed specifically for genomic file compression: quip.

1. Introduction

NGS (Next-Generation Sequencing) sequencing technologies have allowed DNA sequencing at a surprising low speed and costs. Over the past 8 years, the amount of digital memory and network traffic used by sequenced biological data has increased dramatically. Genomic projects such as HapMap, 1000 Genomes, etc., have come to the collection and description of genomes of 2,504 individuals from 26 populations, and they contributed to exponential growth of databases of this type and to the development of increasingly efficient technologies.

Thanks to the large-scale sequencing of samples of DNA, the interest and the new research in these areas by the scientific community are suddenly grown. In a very short time researchers have developed hardware tools, analysis software, algorithms, private databases and infrastructures to support genomics.

In this paper we analyse different approaches for compressing FASTQ, BAM and SAM files generated by Next-Generation Sequencing tools containing nucleotide sequences.

In the first part of this paper we will present the fundamental concepts that are necessary to understand the application domain: i.e. DNA sequencing workflow and the file formats generated by the machines that are currently used for these operations.
In the second part of our work we will discuss and evaluate the compression performance of generic compression tools such as gzip and bzip2 by confronting them with a specific system that was designed specifically for genomic file compression: quip.

2. Next Generation Sequencing Data

The digital processing which leads from the raw reading of DNA sequences to the identification of genetic variants, is a very complex and variable process. The complexity of the workflow is mainly due to possible errors in the reading and alignment phase, which are more likely in the new NGS instruments than in the pre-NGS sequencers.

The continuous evolution in the hardware and software methods and technologies requires frequent changes in the data formats and in the software tools that filter and analyze the sequences. The development of a well-defined and automated workflow for the analysis of genetic data has become of fundamental importance in recent years [2].

A direct result of the increasing amount of DNA readings is the size of the data generated, which continues to present a challenge for the infrastructures that maintain and transmit the data and for the software and statistical analysis algorithms of the sequences. For example, until 2012, more than $13 \times 10^{15}$ nucleotide bases were sequenced and, for a human genome containing about 3 billion nucleotide bases, about 100 gigabytes of data are needed.

NGS (Next-Generation sequencing) is a name chosen to indicate all those sequencing platforms - and the related technologies - born after 2005, which revolutionized the sequencing process, allowing its parallelization, for the benefit of costs and performance.

However, the speed of sequencing for these technologies is at the expense of accuracy in basecalling and alignment, a disadvantage that can be remedied by repeated readings and specific computational processes in the subsequent analysis phases. Basecalling is a process performed by the NGS sequencing tool, which associates with each nucleotide that has been read a probability value for each nitrogenous base. Often the same sequence is read several times to avoid inaccurate readings and unsatisfactory probability values. The most common output data format among NGS platforms is a textual format: FASTQ, of which there are several versions. The Sanger FASTQ version is the de-facto standard, and it is the format accepted by the NCBI Sequence Read Archive for sending the collected data.

With the possibility of sequencing the complete genetic heritage of an individual in just a few days, WGS (Whole Genome Sequencing) approaches are designed to discover genetic variations that contribute to rare or common diseases. Despite the decrease in sequencing costs, these approaches remain very dependent on the management and analysis of a large number of samples [2]. For this reason, alternative methods were born, which focus only on fractions of the entire genome.

The exome is a term derived from Genome and indicates the set of all the exomes present in the genome, or of all the DNA sub-sequences that can encode proteins. The WES (Whole Exome Sequencing) approach is considered a valid alternative to WGS.

The first bioinformatic process in DNA sequence analysis is the alignment process: this is the mapping between the sequences that have been read and the sequence of a reference genome. Many of the alignment software available today are based on two algorithms: the hashed-based method and the Burrows-Wheeler Transform method. The hash-based algorithms use a hash table, based on the reference genome or on the read sequences, to map the set of sequences that have been read in the relative positions of the genome.
More recent algorithms are based instead on string matching by using the Burrows-Wheeler transform (BWT). The BWT algorithms reorder the sequence of the reference genome by grouping into an appropriate data structure the sequences that appear several times. A reference index is then created and is used for rapid placement of the sequences that have been read on the reference genome. The main advantage of the BWT algorithms lies in their speed: they are much faster than the hash-based algorithms.

The next step is the assembly phase, carried out almost always by the same alignment tools. In this phase the sequences that have been read and aligned are now composed together to form the original genetic sequence of the input samples. Also for this phase the reference sequence of the human genome is used, however, by using special overlapping algorithms, it would be possible to reconstruct the original sequence without any reference sequence. This is what happens for example in DNA sequencing for organisms that do not have a genome.

The aligned sequence is stored in SAM (Sequence Alignment / Map) format. At the end of the alignment pipeline the SAM is converted into BAM (binary SAM), a much more compressed format.

The phase following the alignment phase is the trimming of the obtained sequence. In fact, there are several problems arising from sequencing with the NGS machine, mainly due to the alignment of very short sequences. For example, since each short sequence is aligned independently, indel sequences (mutations or recombinations that result in the insertion of an extra codon) may not be aligned with the original sequence. Therefore, before proceeding to subsequent analysis, control processes and improvement of the quality of the readings are carried out.

In this phase we proceed to the extraction of only the exone sequences, that represent a small fraction of the entire genome, but that are believed to code most of the information.

Since sequencing can also capture DNA fragments that originate from non-coding regions, a filter is applied by using a list of known positions for the exone sequences, excluding any reading that does not overlap on these positions.

3. Genomic data file formats
One of the biggest problems in bioinformatics is having to manage a profusion of file formats, often with poorly defined or ambiguous features.

A flat file is a file containing records that have no structural relations. To interpret a flat file you need to know the formatting properties of the file. Flat files for genetic data are delimited ASCII flat files: they can contain a certain number of variables, delimited by special strings (codes). Each code indicates the type of information contained after the code.

By using a file for each sequence, the flat file contains all the reference information, such as description, keywords, bibliographic references related in relation to the work from which the processed data were extracted, position of the sequence in the genome, etc. Some of these flat files, built ad hoc for the needs of the moment, they then became de-facto standards.

The FASTA format is a format for DNA and amino acid sequences. It was originally invented by Bill Pearson as a format for useless tools in the FASTA suite, of his own production [3].

A file in FASTA format begins with a single line describing the sequence. The description lines (defiine) are distinguished from the sequence lines for the “greater” symbol (>) at the beginning of the line. In the definition of the standard it is recommended not to use more than 80 characters for defiines.
The description line usually contains the name or a unique identifier for the sequence, along with various other information. The structure of this header and the type of information it contains are not standardized, but the many database sequences each have their standard FASTA headers. A convention common to these standard headers is that the description line must uniquely identify the next sequence, by using a series of attributes, such as progressive numbers and source databases, that separated by a character.

Over the years FASTQ has become a very common format for sharing genetic data, combining both the sequence of bases and a quality score associated with each nucleic base (i.e. a reliability count on the reading of that base within the sequence). The FASTQ format was born as an extension of the FASTA format, by adding information on the reliability of the reading, thus representing the sequence of sequencing with a higher level of detail, but without weighing on the size of the data, as would happen instead considering the space of the colors or capture image for NGS readings.

Thanks to the extreme simplicity of the format, FASTQ is widely used for the interchange of data, however, even if born as an evolution of FASTA, it also continues to suffer from the absence of a clear and unambiguous definition. A lack that led to the existence of many incompatible FASTQ variants: every NGS sequencing machine produces in output a different FASTQ format.

The SAM (Sequence Alignment / Map format) format is the standard text format for intermediate and final outputs of alignment software. The most common use of these software is to align the input DNA segment (usually a FASTQ file) on a reference genome. The format is a tab-delimited flat file with the @ character for the header lines.

If present, the header must precede the alignment data. Alignment data is organized with an alignment for each line of text. For each alignment line, 11 mandatory fields will be defined, in a predetermined order and delimited by the tab character. If a particular data is not available, the field cannot be left empty, in its place the 0 or * character is inserted (depends on the field).

The BAM format is a coding format for SAM files, compressed in BGZF (Blocked GNU Zip Format) format. BGZF is a block compression format implemented on the gzip standard. BGZF’s goal is to provide good compression along with the ability to access the BAM file in a non-sequential way to perform indexed queries. The BGZF format is compatible with gunzip, which makes it possible to extract a BGZF file by using a gzip tool.

The VCF format [4] is a generic standardized format for storing most existing genetic variants, associated with free annotations.

The VCF file consists of a header section and a data section. The header contains an arbitrary number of metadata, organized on several lines, and a definition line for the structure of the data section. Each metadata line begins with the ## delimiter, while the structure definition line begins with the # character, and the fields defined within it are delimited by tabs.

The meta information in the header can be used to describe the means by which the file was created, the date of creation, the version of the reference sequence, the software used and all the relevant information on the history of the file.
4. Compression of Next Generation Sequencing Data

Figure 1 shows the typical workflow after genome sequencing.

The sequencing process generates a FASTQ file that contains millions of readings generated by the genome. This FASTQ file is subsequently given as input to an alignment program, which generates a SAM file containing all the information, about the alignment and mapping of each reading, based on a reference genome.

Finally, a variant caller analyzes the information contained in the SAM file and finds the differences between the original and the reference genome. The variants found are then stored in a VCF file. With the information contained in the VCF file it is potentially possible to reconstruct the original genome.

Ideally, once all these pipeline steps are completed, only the VCF file should be stored, since it contains all the relevant information about the sequenced genome. This would allow not to memorize the much larger intermediate FASTQ and SAM files.

Unfortunately this is not possible for several reasons. For example the fact that the alignment and the variant calling tools are constantly improved over time and therefore, re-analyzing the raw data might allow in the future the identification of new variants, hitherto not yet found.

However, there are general-purpose compression tools, such as gzip, bzip2 and 7zip, which can be applied directly to any type of genomic data, compressing the size of the file. There are two approaches that are commonly used for the implementation of specific compression tools: i) Compression of assembled genomes and ii) Compression of raw NGS data (ie FASTQ and SAM files).

The compression of the raw sequencing data, that is the content of the FASTQ and SAM files, mainly concerns the compression of identifiers, readings and quality scores. It has been proved that compressing the quality scores is more difficult than compressing the readings, due to the higher entropy rate and the use of a much larger alphabet. If compressed in a lossless way, the quality score size can be up to 70% lower than the original file [5].

4.1. gzip

The FASTQ, SAM, BAM files can be compressed also with generic lossless compression tools. Based on the Lempel-Ziv 77 algorithm (that is widely used in many domains, see for example [6], [7]) gzip is a widely used lossless codec that replaces an archive with a compressed one that has a .gz extension, by maintaining the same properties and access dates.
A .gz file is essentially composed by: A 10-byte extension containing a magic number, program version and timestamp. Optional additional headers (e.g. original file name). A central body, which contains the set of compressed data, and finally 8 bytes containing the CRC-32 type of data checksum and the original length.

Although this file format allows to concatenate multiple input data streams (which are decompressed and concatenated as if they were in a single stream), gzip is usually used to compress a single file.

4.2. bzip2

bzip2 is a lossless compression algorithm based on the Burrows-Wheeler transform and Huffman coding. The encoding is done by compressing the file in blocks each independent of the other.

Among the various features there is a tool, bzip2recover, that retrieves damaged archives: if, during transmission, a file is damaged, you can try to recover the data from the undamaged blocks by checking the CRC-32 control strings of each single block.

4.3. quip

Quip [8] is a special purpose lossless compressor targeted to the compression of genetic data. It is based on arithmetic coding (another compression strategy that is used in many domains, see for example [9], [10], [11]).

In quip, the same arithmetic encoder is used to encode quality scores, identifiers of readings, nucleotide sequences and information about alignment, but the codec uses very different statistical models depending on the type of data it deals with, and this provides a major advantage over general purpose compression algorithms. Furthermore, all parts of the algorithm use adaptive modeling, which means that all the coding parameters are trained and updated at run time while data is compressed, so as to increase the compression rate on large files.

There are many data specific improvements to the simple arithmetic coding. A reading identifier is unique for each reading. Actually, for the management of them, even an integer would be enough to increase with each reading, but typically each reading has a complex string containing the name of the instrument, identifier of the execution, identifier of the flow cell and the coordinates of the filament. Most of this information is the same for each reading and is simply repeated, going to inflate the file size. To remove this redundancy, a form of delta encoding is used in quip. A parser separates the ID into separate fields which are compared to the previous ID. Tokens that remain unchanged from one reading to another (for example the name of the instrument) can be compressed using, in some cases, less than 1 bit. The numeric tokens are recognized and memorized efficiently, both directly and as an offset between the current one and the token that is at the same position in the previous reading. Non-identical tokens are encoded by trying to match the prefix of the previous encoded token. As a final result of compression, the IDs of the readings will be stored in 2-4 bytes, at the front of the 50 bytes or more of the original dimensions.

To compress the nucleotide sequences, a simple model based on Markov chains is used. The nucleotide in a given position in the reading is predicted by using the 12 previous positions. This model uses more memory than the classic general-purpose algorithms ($4^{13} = 67108864$ parameters are required, each one represented with 32 bits) but it is simpler and extremely efficient (a small computation is required and the execution time is limited mainly by the memory latency).

The quality score in a given position is closely related to the one in the previous position. This makes a Markov chain a natural model for determining the probabilities associated with the arithmetic
encoder, but, compared to nucleotides, the quality scores use a much larger alphabet and this limits the order of the Markov chain and its precision.

quip also provides lossless compression based on references. Given as input some readings aligned in SAM or BAM format and the reference sequence (not aligned), in FASTA format, the readings are compressed by preserving all the information of the SAM / BAM file, including the header, the reading identifiers, information about the alignment and all the optional fields in the SAM format. Unaligned readings are stored and compressed by using the Markov chain model.

quip is a command line application. The file that quip generates as an output has the extension .qp.

5. Experimental results

We have tested quip, gzip e bzip2 on a set of next generation sequencing data files often used in this field, that are available over the internet: (http://genome.crg.es/encode_RNA_dashboard/hg19 and ftp://ftp.ncbi.nlm.nih.gov/hapmap).

Our test dataset is composed by files in FASTQ or SAM/BAM formats. Table 1 summarizes the specifications of our test data set.

| Sequences          | Description | Format | Size  |
|--------------------|-------------|--------|-------|
| GM-12878-CYTOSOL   | Short RNA   | FASTQ  | 5.31 GB |
| GM-12878-CYTOSOL   | Short RNA   | BAM    | 29.1 MB |
| GM-12801S4         | Repli-Seq Illumina | FASTQ | 1.01 GB |
| BG02-1-DS9028-FC20EM7-1 | Repli-Seq Illumina | FASTQ | 739.6 MB |
| BG02-1-DS9028-FC20EMB-6 | Repli-Seq Illumina | FASTQ | 797.3 MB |
| NA12156 (No graph) | CEPH/UTAH PED. 1408 | BAM | 8.8 MB |
| GM-06990S3         | Repli-Seq Illumina | BAM | 288.3 MB |

Figure 2 shows the results of applying the gzip, bzip2 and quip tools to our test data set.

In the figure the bzip2 compression results, in terms of size of the lossless compressed file, are outlined by the yellow lines, the gzip results by the grey line and the quip results by the orange.

The blue line represents the uncompressed file size.

All these tools achieve a compression ratio that in average is close to 70%.

In particular gzip has the worst performance and bzip2 is close, but almost always outperformed, to quip.

The compression performances do not seem to depend on the file format, and the compression results seem stable independently on the fact that the compressed file is a FASTQ or a BAM.
Figure 2. Testing results
6. Conclusions
In this paper different approaches for the compression of FASTQ, BAM and SAM files generated by Next-Generation Sequencing tools containing nucleotide sequences have been analyzed.

In the first part of the work the basic concepts of the application domain have been presented: DNA sequencing workflow and the file formats generated by the machines used for these operations. In the second part of our work we discussed and evaluated generic compression tools, namely gzip and bzip2, together with a specific system compression system targeted to these genetic file formats: quip.

The experimental evidences have shown that quip is very effective and outperforms gzip and bzip2 on almost all the files in our test data set.

Future work involves a deeper experimentation and the design of better specific compressors.

Acknowledgment
The author thanks his student Felice D’Avino for conducting preliminary tests on the compression of FASTQ and SAM/BAM files.

References
[1] D. C. Koboldt, L. Ding, E. R. Mardis, and R. K. Wilson, “Challenges of sequencing human genomes”, Briefings in Bioinformatics, vol. 11, pp. 484-498, 2010.
[2] D. J. Hedges, D. Burgess, E. Powell, C. Almonte, J. Huang, S. Young, B. Boese, M. Schmidt, M. A. Pericak-Vance, E. Martin, X. Zhang, T. T. Harkins, and S. Zuchner, “Exome Sequencing of a Multigenerational Human Pedigree”. PLoS ONE, 4:8, 2009.
[3] W. R. Pearson and D. J. Lipman, “Improved tools for biological sequence comparison”. Proceedings of the National Academy of Sciences of the United States of America, 85(8):2444-8, 1988.
[4] P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. De Pristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, G. McVean and R. Durbin, “The variant call format and VCF tools”. Bioinformatics, 27, pp. 2156-2158, 2011.
[5] J. K. Bonfield and M. V. Mahoney, “Compression of FASTQ and SAM format sequencing data”. PloS one, 8(3):e59190, 2013.
[6] F. Rizzo, J. A. Storer, and B. Carpentieri. “ LZ-based image compression”. Information Sciences 135, no. 1-2, pp. 107-122, 2001.
[7] R. Pizzolante and B. Carpentieri, “Visualization, band ordering and compression of hyperspectral images”. Algorithms 5, no. 1, pp. 76-97, 2012.
[8] D. C. Jones, W. L. Ruzzo, X. Peng, M. G. Katze, “Compression of next-generation sequencing reads aided by highly efficient de novo assembly”. Nucleic Acids Research, Vol. 40, no.22. 2012.
[9] R. Pizzolante and B. Carpentieri, “Lossless, low-complexity, compression of three-dimensional volumetric medical images via linear prediction”, Proceedings of 18th International Conference on Digital Signal Processing (DSP), 2013.
[10] R. Pizzolante, A. Castiglione, B. Carpentieri, A. De Santis, and A. Castiglione. “Protection of microscopy images through digital watermarking techniques”. Proceedings of the international conference on intelligent networking and collaborative systems, IEEE INCoS 2014; 2014.
[11] A. Castiglione, R. Pizzolante, F. Palmieri, B. Masucci, B. Carpentieri, A. De Santis, and A. Castiglione. “On-board format-independent security of functional magnetic resonance images”. ACM Transactions on Embedded Computing Systems 16, no. 2, 2017.