SOLiDzipper: A High Speed Encoding Method for the Next-Generation Sequencing Data

Young Jun Jeon¹, Sang Hyun Park¹, Sung Min Ahn² and Hee Joung Hwang¹

¹SDLAB, Gachon University of Medicine and Science, 406-799 Yeonsu-dong, Incheon, Korea. ²Laboratory of Genomics and Genomic Medicine, Lee Gil Ya Cancer and Diabetes Institute, Gachon University of Medicine and Science, Incheon, Korea. Corresponding author email: smahn@gachon.ac.kr; hwanghj@gachon.ac.kr

Abstract

Background: Next-generation sequencing (NGS) methods pose computational challenges of handling large volumes of data. Although cloud computing offers a potential solution to these challenges, transferring a large data set across the internet is the biggest obstacle, which may be overcome by efficient encoding methods. When encoding is used to facilitate data transfer to the cloud, the time factor is equally as important as the encoding efficiency. Moreover, to take advantage of parallel processing in cloud computing, a parallel technique to decode and split compressed data in the cloud is essential. Hence in this review, we present SOLiDzipper, a new encoding method for NGS data.

Methods: The basic strategy of SOLiDzipper is to divide and encode. NGS data files contain both the sequence and non-sequence information whose encoding efficiencies are different. In SOLiDzipper, encoded data are stored in binary data block that does not contain the characteristic information of a specific sequence platform, which means that data can be decoded according to a desired platform even in cases of Illumina, Solexa or Roche 454 data.

Results: The main calculation time using Crossbow was 173 minutes when 40 EC2 nodes were involved. In that case, an analysis preparation time of 464 minutes is required to encode data in the latest DNA compression method like G-SQZ and transmit it on a 183 Mbit/s bandwidth. However, it takes 194 minutes to encode and transmit data with SOLiDzipper under the same bandwidth conditions. These results indicate that the entire processing time can be reduced according to the encoding methods used, under the same network bandwidth conditions. Considering the limited network bandwidth, high-speed, high-efficiency encoding methods such as SOLiDzipper can make a significant contribution to higher productivity in labs seeking to take advantage of the cloud as an alternative to local computing.

Availability: http://szipper.dinfree.com. Academic/non-profit: Binary available for direct download at no cost. For-profit: Submit request for for-profit license from the web-site.

Keywords: bioinformatics, NGS, DNA compression, cloud computing

Evolutionary Bioinformatics 2011:7 1–6
doi: 10.4137/EBO.S6618

This article is available from http://www.la-press.com.

© the author(s), publisher and licensee Libertas Academica Ltd.

This is an open access article. Unrestricted non-commercial use is permitted provided the original work is properly cited.
Introduction

Next-generation sequencing (NGS) methods, which are revolutionizing genomics research by reducing sequencing cost and increasing its efficiency,\(^1\) pose various computational challenges of handling large volumes of short read data. For example, human genome re-sequencing at ~30X sequencing depth requires a level of computational power achievable only via large-scale parallelization.\(^2\)

One potential solution to these computational challenges is the use of cloud computing. Langmead and colleagues\(^3\) genotyped data comprising 38-fold coverage of the human genome in ~4 h on the Amazon cloud (Amazon EC2) using the Crossbow genotyping program. In a recent study, Parul Kudtarkar and colleagues\(^4\) computed orthologous relationships for 245,323 genome-to-genome comparisons on the Amazon cloud using the genomic tool, Roundup, at a lesser cost. Applied Biosystems provides a cloud computing service as an alternative to maintaining an in-house computing infrastructure for NGS data analysis (ie, SAMtools\(^5\)) to the SOLiD system users (ABI SOLiD system). Despite the promises and potential of cloud computing, the biggest obstacle to moving to the cloud may be network bandwidth, since it may take at least a week to transfer a 100 gigabyte NGS data file across the internet in a typical research environment.\(^6\)

The more dramatically the advantages of NGS data sequencing or analysis in a cloud environment are revealed, the more apparent will the limitations of access to a cloud environment become. For example, we are currently working on the Amazon cloud wherein we can control the number of nodes needed for the analysis at our chosen time and predict the cost required for the analysis operation. However, we cannot say that our chosen transmission time will guarantee optimal bandwidth when transmitting large volumes of NGS data to a cloud environment. In addition, the usable bandwidth in a lab is a limited resource. Thus, we can decrease the rate of offsetting time benefit, one of the many benefits during an experiment on a cloud, by applying a proper encoding method and using a transmission bandwidth efficiently to transmit NGS data. Furthermore, the important points that we should take into consideration are the encoding/decoding time and the possibility of parallel/selective decompression as well as the compression rates when adopting an encoding method aimed at cloud transmission unlike the traditional DNA compression method aimed at efficient storage.

Efficient encoding methods may enable to overcome the problems of transferring such a large dataset. Recently, Tembe and colleagues\(^7\) showed that NGS data can be reduced by 70%–80% in size by using their algorithm. However, when a large dataset is encoded for transferring, the time required for encoding and decoding is equally as important as the encoding efficiency. Accordingly, an ideal compression algorithm that can be used in combination with cloud computing for sequence data analysis needs to have the following features: 1) high encoding/decoding rate; 2) high encoding efficiency; 3) a parallel technique to decode and split compressed data in the cloud.

Here, we present SOLIDzipper, a new encoding method, by which we can encode NGS data with high speed and high efficiency. SOLIDZipper is best optimized to encode csfasta and QV files from ABI SOLiD system.

Methods

The basic strategy of SOLIDZipper is to divide and encode. NGS data files contain both the sequence and non-sequence information whose encoding efficiencies are different. In SOLIDZipper, the non-sequence information including the sequence IDs and number in plain text format is encoded by a general purpose compression algorithm (ie, gzip, bzip2, lzma(LZMA SDK)), whereas the sequence information consisting of ‘0123’ in csfasta format, which has random patterns and thus a low encoding efficiency, is encoded by bitwise and shift operations. Figures 2 and 3 summarize the encoding process and the method of SOLIDzipper, respectively.

Decoding of SOLIDZipper is basically a reverse of encoding, except for non-calls. In SOLIDZipper, non-calls (‘.’ in csfasta files) are converted into temporary binary data when encoded. When decoded, QV values are used in order to recover temporary binary data to previous non-calls.

Unlike other general encoding methods, SOLIDZipper does not use compression dictionary scheme or statistical pattern matching (ie, palindromes, string comparisons, repeat detection, data permutation),\(^7\)–\(^11\) thereby minimizing computing resource requirements.
SOLiDzipper: A high speed encoding method for the NGS

and dictionary exploring time. For example, G-SQZ\textsuperscript{7} utilizes Huffman coding\textsuperscript{12} method, which generates a Huffman tree in the process of highly efficient encoding. And the DNA Compress\textsuperscript{10} program shows fast and effective encoding using detection of repeats. Such a dictionary exploring time or statistical pattern matching time used in the computational method can require a significant amount of time for the encoding process that should not be ignored when processing huge volumes of NGS data. Thus, it will be more effective to complete encoding as fast as possible even by lowering the encoding rate a little when transferring data to cloud computing for high-performance sequence analysis. SOLiDzipper performs high-speed, high-efficiency encoding on the bitwise level by taking advantage of the characteristic features of NGS data.

SOLiDzipper is implemented in Java 1.6 command-line mode at 64 bit Linux machine (Linux: 2.6.29.4-167.fc11.x86_64 Fedora 11 64 bit, Intel(R) Core(TM) 2 Duo CPU E8400 3.00GHz, 4 GByte memory).

Figure 1. Rt changes according to the data transfer speed and encoding methods. Notes: When data transfer speed across the internet exceeds a certain threshold, it offsets the advantages of encoding NGS data. For example, LZMA does not provide any advantage when the transfer speed is 10 megabytes per second. Within the current limitations of data transfer speed, SOLiDzipper shows the best performance among the algorithms compared, providing a definite advantage in Rt. X axis (logarithmic scale); Y axis (linear scale).

Figure 2. Encoding process of SOLiDzipper. Notes: Sequence IDs are extracted from QV and csfasta files and then are bitwise-encoded (A and B). Extracted sequence IDs are combined and compressed using the general purpose compression methods (eg, gzip) (C). Encoded data are stored in a data block (D).
Table 1 shows the comparison experiments for zipping using high speed zipping option (—fast) of general purpose compression tool gzip (version 1.3.12), highest zipping efficiency option (mx = 9) of LZMA (version 4.65) (LZMA SDK) and G-SQZ (version 0.6). 133 gigabytes of mate-paired data from the ABI SOLiD 3.5 system were used as the test data set.

Results and Discussion

Encoding efficiency is usually regarded as the most important criterion to determine the performance of encoding algorithms, especially when it is used to reduce the long-term storage cost. However, when encoding is used in combination with cloud computing, NGS data need to be encoded in the local servers and then decoded in the cloud as quickly as possible (ie, in this case, encoding is used to facilitate transfer, not for long-term storage).

When cloud computing is used for NGS data analysis, ready to job time (Rt) represents the sum of time required for compression in the local servers, transferring the compressed data to the cloud and decompression in the cloud. Rt increases in proportion with the increase in time required for compression and decompression, thereby offsetting the advantages of using cloud computing for higher efficiency.

The Ready to job time (Rt) of cloud computing for NGS data analysis can be calculated using equation below (1).

\[
R_t = \text{Encode}(\text{NGS data})t + \left(\frac{\text{size of encoded NGS data}}{\text{data transfer speed}}\right)t + \frac{\text{Decode}(\text{encoded NGS data})}{\text{decoding unit count}}
\]

Table 1. Comparison of encoding efficiencies and time between the different encoding methods (decoding unit count = 1).

| 133 GBytes of csfasta and QV files | Compression time (minutes) | Decompression time (minutes) | Compression rate |
|-----------------------------------|---------------------------|-------------------------------|-----------------|
| SOLiDzipper                       | 61                        | 62                            | 74.1%           |
| gzip (—fast)                      | 60                        | 54                            | 64.9%           |
| lzma (mx = 9 ultra)               | 3640                      | 72                            | 77.0%           |
| G-SQZ                             | 177                       | 571                           | 77.1%           |
When time factor is considered, the advantages of using encoding methods are offset when the data transfer speed exceeds a certain threshold (Fig. 1). However, within the current limitations of data transfer, SOLiDzipper is more time-efficient than both gzip (low compression rate and high operation speed) and G-SQZ (high compression rate and low operation speed).

In addition, SOLiDzipper does not use compression scheme, generating data blocks of the same length. Since there is no link between the compressed data blocks, encoded data can be distributed for parallel decoding, thereby drastically enhancing the decoding rate in the cloud.

The contributory point of SOLiDzipper to bioinformatics was to address the issue of high-speed transmission infrastructure that could not expand easily at a lower cost, which can be achieved with a DNA analysis environment in a cloud environment like Amazon EC2.

The objective of SOLiDzipper is to minimize the percentage of preparation time in the entire DNA analysis process time so that the analysis environment can smoothly move to the cloud, rather than to merely increase the encoding speed or compression efficiency.

We divided the entire processing time into two parts; the first part is the preparation time for analysis and represents the time required to compress DNA data produced on a sequence platform, transmit them over a network, and decode them on a cloud; and the other is the main computation time that represents the time required to carry out computation analysis on a cloud in parallel.

Table 2 presents the calculations of the required time until the final analysis results considering the communication bandwidth and data compression method based on the whole-genome computation time of Crossbow.

According to the main computation time of Crossbow, 10 workers took less than 7 hours to compute the whole genome and 40 workers achieved the same within 3 hours in the Amazon cloud environment. In addition, it took more than an hour to transmit (183 Megabit/second transfer speed) the compressed data set (103 GigaByte) to Amazon s3. In a situation where the transmission bandwidth is limited the data compression time should be considered, which raises an important issue since more time is required to prepare the analysis than the actual analysis time in a cloud environment. For example, a data set of about 300 GB compressed using G-SQZ with high compression rate, is transmitted through 45 Megabits bandwidth, and is decompressed in parallel at 40 nodes to conduct the Crossbow analysis. In such a case, it takes roughly three times to prepare the operation

| Encoding method | Transfer speed (Megabit/s) | EC2 nodes (workers) | Processing time (minute) | Encoding | Transfer | Parallel decoding | Crossbow computation | Total |
|----------------|---------------------------|---------------------|--------------------------|---------|----------|------------------|----------------------|-------|
| Gzip           | 45                        | 10                  | 137                      | 306     | 5        | 390              | 838                  |       |
|                | 40                        |                     | 137                      | 306     | 1        | 173              | 617                  |       |
|                | 183                       | 10                  | 137                      | 77      | 5        | 390              | 609                  |       |
|                |                           | 40                  | 137                      | 77      | 1        | 173              | 388                  |       |
| G-SQZ          | 45                        | 10                  | 399                      | 200     | 57       | 390              | 1047                 |       |
|                | 40                        |                     | 399                      | 200     | 14       | 173              | 787                  |       |
|                | 183                       | 10                  | 399                      | 50      | 57       | 390              | 896                  |       |
|                |                           | 40                  | 399                      | 50      | 14       | 173              | 637                  |       |
| SOLiDzipper    | 45                        | 10                  | 135                      | 226     | 6        | 390              | 758                  |       |
|                |                           | 40                  | 135                      | 226     | 2        | 173              | 536                  |       |
|                | 183                       | 10                  | 135                      | 57      | 6        | 390              | 588                  |       |
|                |                           | 40                  | 135                      | 57      | 2        | 173              | 367                  |       |

Notes: Encoding time was calculated by assuming that the dataset was 300 GigaByte and applying the encode time and compression rate from Table 1. The parallel decode time was obtained by dividing the decode time from Table 1 by the number of nodes at EC2 based on the assumption that the decode operation would be performed in a cloud environment. The transfer speed of 183 Megabit/second was used during uploading in the Crossbow computation. In addition, 1/4th of 183 Mb/s was also used in the calculation of transfer speed just like 1/4th of the 40 workers were used in the Crossbow computation.
than the actual computation time in the cloud. Thus, compression time should be considered important in addition to compression efficiency when considering transmission to a cloud environment.

These findings indicate that the entire processing time can be reduced according to the encoding methods used, if the same communication bandwidth is adopted. Considering the limited network bandwidth, high-speed, high-efficiency encoding methods such as SOLiDzipper can make a significant contribution to higher productivity in labs seeking to take advantage of the cloud as an alternative to the local computing cluster.

Conclusions
The unique features of SOLiDzipper are: 1) It divides information in csfasta files for high encoding efficiency and speed; 2) It combines two different compression methods (i.e., bitwise/shift and general purpose compression), allowing for the optimal preparation time (Rt) for Cloud Computing; 3) Data can be decoded selectively without unzipping the whole encoded file because encoded data are stored as data blocks of the fixed size; 4) In the cloud, encoded data can be distributed for parallel decoding; 5) It requires minimal computing resources.

In summary, SOLiDzipper is a fast encoding method that can efficiently encode and decode NGS data. This method can be especially more suited to typical research environments where the data transfer speed across the internet is limited.

Disclosure
This manuscript has been read and approved by all authors. This paper is unique and is not under consideration by any other publication and has not been published elsewhere. The authors and peer reviewers of this paper report no conflicts of interest. The authors confirm that they have permission to reproduce any copyrighted material.

References
1. Metzker ML. Sequencing technologies the next generation. Nature Reviews Genetics. 2010;11:31–46.
2. Ahn SM, et al. The first Korean genome sequence and analysis: full genome sequencing for a socio-ethnic group. Genome Research. 2009;19:1622–9.
3. Langmead B, et al. Searching for SNPs with cloud computing. Genome Biology. 2009;10:R134.
4. Parul Kudtarkar, et al. Cost-Effective Cloud Computing: A Case Study Using the Comparative Genomics Tool, Roundup. Evolutionary Bioinformatics. 2010;6:197–203.
5. Li, et al. The Sequence Alignment/Map format and SAMtools. Bioinformatics. 2009;25:2078–9.
6. Stein LD. The case for cloud computing in genome informatics. Genome Biology. 2010;11:207.
7. Tembe W, et al. G-SQZ: Compact Encoding of Genomic Sequence and Quality Data. Bioinformatics. 2010;26:2192–4.
8. Adjeroh, et al. DNA sequence compression using the burrows-wheeler transform. Proc IEEE Comput Soc Bioinfom. 2002.
9. Brandon, et al. Data structures and compression algorithms for genomic sequence data. Bioinformatics. 2009;25:1731–8.
10. Chen X, et al. DNACompress: fast and effective DNA sequence compression. Bioinformatics. 2002;18:1696–8.
11. Soliman, et al. A lossless compression algorithm for DNA sequences. Int J Bioinf Res. 2009;5:593–602.
12. Huffman DA. A Method for the Construction of Minimum-Redundancy Codes. Proceedings of the IRE. 1952;40:1098–102.

Publish with Libertas Academica and every scientist working in your field can read your article

“I would like to say that this is the most author-friendly editing process I have experienced in over 150 publications. Thank you most sincerely.”

“The communication between your staff and me has been terrific. Whenever progress is made with the manuscript, I receive notice. Quite honestly, I’ve never had such complete communication with a journal.”

“LA is different, and hopefully represents a kind of scientific publication machinery that removes the hurdles from free flow of scientific thought.”

Your paper will be:
• Available to your entire community free of charge
• Fairly and quickly peer reviewed
• Yours! You retain copyright

http://www.la-press.com