International Journal of Control and Automation
Vol.8, No.8 (2015), pp.223-232
http://dx.doi.org/10.14257/ijca.2015.8.8.23

SolidStream: NGS Encoding Utility with High Speed and High Efficiency Considering Transmission Bandwidth of Cloud Environments

Seokjin Im[1], HeeJoung Hwang[2*] and Jinsong Ouyang[3]

[1]*Dept. of Computer Engineering, Sungkyul University, 53 Sungkyuldaehak-Ro, Manan-Gu, Anyang-Si, Gyeonggi-Do, 430-742, Korea*
[2]*Dept. of Computer Engineering, Gachon University, 1342 Seongnamdae-Ro, Sujeong-Gu, Seongnam-Si, Gyeonggi-Do, 461-701, Korea*
[3]*Dept. of Computer Science, California State University Sacramento, 6000 J Street Sacramento, CA, 95819, U.S.A.*
[1]*imseokjin@gmail.com,* [2]*hwanghj@gachon.ac.kr,* [3]*jouyang@csus.edu*
[*]*Corresponding Author*

Abstract

Cloud computing implemented in conjunction with next generation communication technologies has brought about many changes in the field that utilizes enormous computing power. Especially in the field of biomedical sciences, the computing environment allows for quick analysis of next-generation sequencing (NGS) data by providing flexible and nearly unlimited computing resources. In cases where data analysis requires huge computing power, large amounts of data should transmit from a local cluster to the cloud via the next generation communication network. However, the limited bandwidth in the network can cause slow transmission speeds and connection delays. These limitations may be a serious obstacle for efficient data analysis. In order to resolve the obstacle, we propose SolidStream, a method that improves the transmission of NGS data to the cloud. The proposed SolidStream adopts a strategy of simultaneously encoding and transmitting NGS data. In SolidStream, NGS data is encoded in blocks, and each block is linked immediately to the transmission stream. Furthermore, SolidStream manages the encode stream and the transmission bandwidth using a linear buffer to improve the throughput. We evaluate the performance of SolidSteam for NGS data encoding and transfer against that of existing algorithms. When compared to gzip compression, SolidStream reduces the time needed for compression and transmission of NGS data to the cloud by a factor of 4. When compared to an NGS compression method, SolidStream reduces the time by a factor of 10. The results of the evaluation indicate that SolidStream enables efficient analysis NGS data in the cloud.

Keywords: *Next-Generation Sequencing data, Cloud computing, Data Compression*

1. Introduction

Next-generation sequencing (NGS) produces high-quality biological data that improves data analysis in bio-medical fields. A dielectric analysis with Solexa technology of a variety of DNA sequencing platforms consumes hundreds of gigabytes of space. The large amount of data can be an obstacle to effective analysis. Therefore, in order to process and analyze biological data with consistency and at a lower cost, we require a cluster of powerful arithmetic processing units equipped with large storage space [1, 2].

Cloud computing can eliminate this obstacle by offering large quantities of computing power and storage that dramatically reduce the time needed from hundreds of hours to just several hours [3, 4]. Cloud service providers (e.g., Amazon Elastic Compute Cloud

ISSN: 2005-4297 IJCA
Copyright ⓒ 2015 SERSC

and Tsinghua Cloud) offer various bioinformatics software tools that users can customize [5]. For example, Life/SOLiD, a platform for NGS platforms, takes 7 to 14 days to produce approximately 30GB to 50GB of data in a run, and Illumina-Solexa takes 4 to 9 days to produce approximately 18GB to 35GB of data in a run. After the data is generated, we must transmit that data to the server provided by the cloud service provider. Then, the cloud services can allow us to compute the results of the analysis.

Limited bandwidth and connection delays, however, can be bottlenecks to efficient analysis when a large amount of NGS data is transmitted to the cloud. As a way to resolve the bottlenecks, several efficient data compression methods have been introduced to specially reduce the size of the NGS data that is transmitted. These methods have a high-compression speed and compression ratio because general compression methods cause a temporal cost of dozens of hours per large data set.

According to Langmead et al. [3], the processing time for arithmetic analysis operations in the cloud were of 173 minutes when using 40 EC2 nodes. However, the time elapsed from the beginning of the compression of the NGS data until the completion of the transmission to the cloud (R_t) was of 614 minutes when using G-SQZ, a modern DNA compression method that operates at a network bandwidth of 45 Megabits/second [7].

Thus, we can increase the efficiency of the transmission of a large data sequence to the cloud by adopting improved compression algorithms. However, the efficiency can be further improved for cases in which the data sequence is transmitted after it is completely compressed.

In this paper, we present a solution for the problem where data is transmitted only after it is completely compressed. We propose SolidStream, a method that integrates the compression and the transmission in order to improve the efficiency of the transmission to the cloud. SolidStream concurrently executes the transmission and the compression by compressing NGS data in blocks and then forwarding each generated block to the upload stream to be transmitted to the cloud. Thus, SolidStream compresses and transmits data simultaneously and reduces the total time needed for the compression and the transmission when compared to existing algorithms that transmit data after it is completely compressed.

The rest of the paper is organized as follows. We present SolidStream, the proposed method, in Section 2. Next, we show the results of the performance evaluations in Section 3, and we discuss those results. In the last part we conclude the paper in Section 4.

2. The Proposed SolidStream

SolidStream is designed by integrating the compression and transmission into one process in order to reduce the entire time needed to forward the compressed NGS data to the cloud. Figure 1 shows how SolidStream processes NGS by transmitting data through an upload stream immediately after it is compressed through the integrated encoding method. In order to improve the performance, the throughput of the upload stream and the transmission bandwidth are actively linked to the speed of the compression process. SolidStream compresses and transmits NGS data to the cloud in the sequence of steps that are provided as follows in order for that data to be analyzed by the cloud server.

 Copyright ⓒ 2015 SERSC

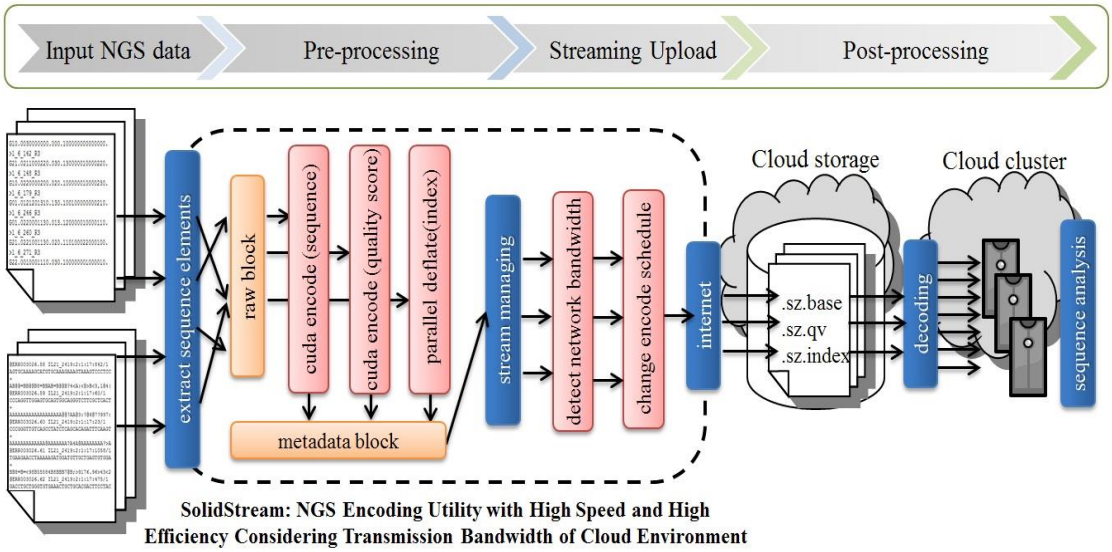

Figure 1. The Workflow of SolidStream

2.1. Pre-processing Step

In this step, the raw data are basically disassembled and then are compressed in metadata blocks. SolidStream adopts SolidZipper, our previous work, as a basic compression algorithm. This algorithm has a higher compression rate for the NGS data than other algorithms, including gzip (which uses deflate algorithm) and G-SQZ (which uses the Huffman algorithm) [6]. SolidZipper uses the csfasta and fastaq characteristics of the DNA sequence format and separates additional information (base, quality value, and ID) from the NGS sequences. The speed and rate of compression is improved by using encoding method that is appropriate for each property of the separated data.

To further enhance the speed of the compression higher than SolidZipper, we implement the basic SolidZipper algorithm in CUDA as a multi-threaded parallel process. SolidZipper compresses the NGS data in blocks and arranges the operations of parallel arithmetic into the appropriate threads in CUDA, and maintaining balance between the operations.

As shown in Figure 1, SolidStream divides the raw NGS data into raw byte data and a block reference list and then compresses the separated data by encoding into metadata blocks through SolidZipper .

Figure 2 shows the activity diagram of the compression and transmission of NGS data in SolidStream. In block B-1, SolidZipper encodes the NGS data in metadata blocks with high speed and efficiency and forwards those data blocks to block B-2.

In B-1 of Figure 2, the encoding administrator generates a reference list of the byte data that is needed when each thread reads the data to improve the coalescing of GPU job processing as in Figure 3, before loading the CUDA operations. When CUDA arithmetic starts, each thread refers to a reference list loaded onto the shared memory and begins the encoding process. When the CUDA arithmetic operation is finished, the CUDA job loader writes the encoded data to the stream buffer of B-1-a, as shown in Figure 4, and performs preprocessing for the next operation.

Copyright ⓒ 2015 SERSC

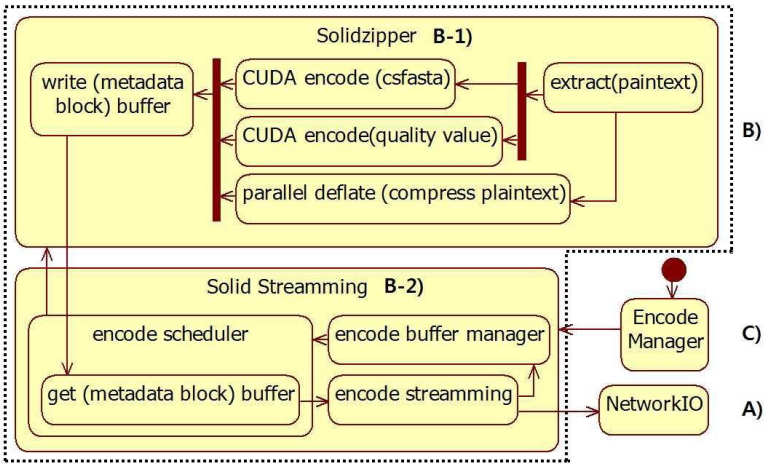

Figure 2. The Workflow of SolidStream

Figure 3 shows the SolidStream encoding process with parallel CUDA processing in detail. Here, T_1, T_2 … T_n indicate the CUDA threads. Each thread has 2450 bytes of throughput and 7 block references. Each thread divides a byte block into 7 sub-blocks and allocates 50 base-sequences per sub-blocks.

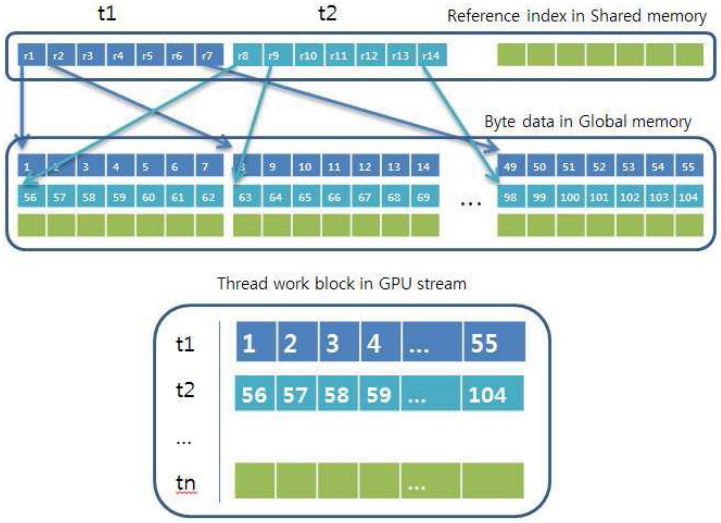

Figure 3. The Encoding Scheme with SolidZipper Applied

As soon as each metadata block is generated, it is forwarded to the next step for the transmission to the cloud, unlike in our previous work where we used SolidZipper as a standalone program [6]. Thus, the integration of the compression and transmission in SolidStream helps the performances improved much more than using SolidZipper standalone separately from the transmission like in the previous work [6].

2.2. Streaming Upload Step and Post-processing Step

SolidStream connects to an upload network stream to transmit each compressed metadata block to the cloud. Block B-2 in Figure 2 is responsible for conducting the stream transmission of the data encoded at B-1.

Copyright ⓒ 2015 SERSC

Figure 4 shows the stream buffer manager in more detail. B-2-d receives outside events to change the encoding cycles, while B-1-a, which is based on the cycles designated by B-2-d, encodes the NGS data to be written inside the memory buffer. At the same time, the encoding streaming site reads the written data to the memory buffer for it to be transmitted to the network. B-2-c computes the slope of the speed of the buffer accumulation. Then the event signals are changed by the encoding schedule cycle and are then transmitted to the scheduler.

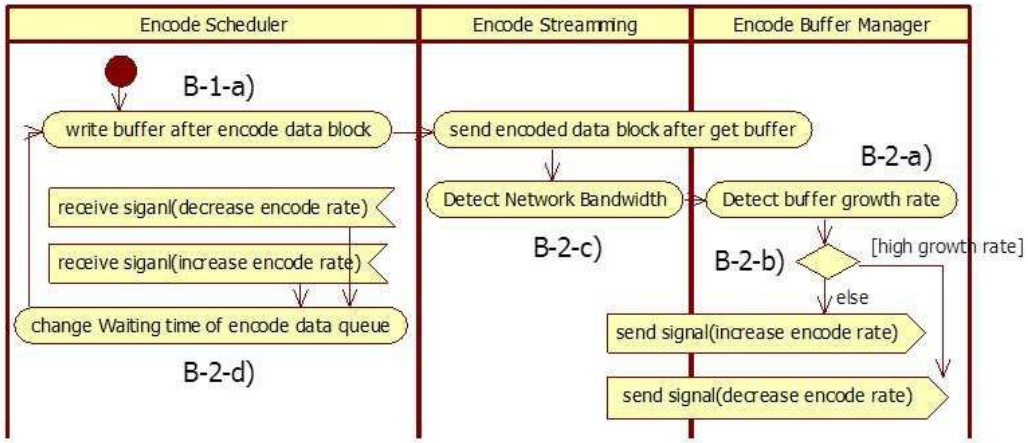

Figure 4. The Stream Buffer Manager

Finally, in the post-processing step, the cloud server stores the compressed data, and then proceeds to decode and analyze the data.

3. Performance Evaluation

3.1. Implementation

In order to evaluate the performance of SolidStream, we implemented it in Java 1.6 command line mode on a 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], Apache MINA 2.0.4, CUDA 4.2 (build 9), and Java bindings for CUDA (JCUDA) 0.4.2.

SolidStream core (Figure 2) is organized with job collaboration through Compute Unified Device Architecture (CUDA) [14] encoding part shown in Figure 3 and the job scheduling part (Figure 4). It is located within the IO filter chain of an Apache MINA [15]. Apache MINA offers a framework for easily developing high-performance and high-scalability network applications. It enables various encoding methods that can be selectively applied within the IO Filter Chain or can be applied through a connection with the chain. When a specific parameter is given as the NGS data in the driving client, the IO Handler encodes the data by reading the raw data in a fixed block size. With the initial option where the client conducts the encoding, the data sent by the designated encoder are transmitted to the server through a communication session, as seen in part A of Figure 2.

In order to prove that SolidStream has a high compression rate and that it transmits the compressed data quickly to the cloud, we compare SolidStream to existing compression algorithms, *i.e.*, gzip, pigz, g-sqz and SolidZipper.

While SolidZipper runs on local IO in a standalone mode in this study, the current network bandwidth should be considered when conducting IO output on the network IO, as seen in part A of Figure 2. Considering the network bandwidth and the encoding

Copyright ⓒ 2015 SERSC

throughput, the buffer overflow between the client and server can be prevented by controlling the encoding schedule, as shown in Figure 4.

3.2. Results and Discussion

We present the time elapsed from the beginning of the compression of the NGS data until the completion of the transmission to the cloud (R_t) to evaluate the performance of SolidStream.

Table 1 depicts the values of R_t in seconds with respect to various bandwidth values from 1Mbps to 250Mbps when the input is 66.7 GByte of csfasta NGS data. Here, 2C indicates the simulation condition where an Intel dual core 2.80GHz, Windows7 32bit, and 4GB memory are used, and 8C means the simulation condition in which Intel i7-2600k 8core 3.40GHz, Linux Fedora11 64bit, and 16GB memory are used. Also, in LG, GeForce 9500 GT and Compute Capability 1.1 are used.

The table shows that SolidStream compresses data and transmits data more quickly than the existing schemes. In general, a buffer underflow may temporally sever the video stream of users in stream transmission of video images. The ideal situation in a stream transmission involves the accordance of encoding throughput and network transmission bandwidth. However, when the encoding throughput exceeds the amount that can be accepted by the transmission of the network bandwidth, the encoding schedule is controlled, thereby preventing a buffer overflow by a transmission source with many gigabytes of encoded data. In contrast, when the encoding throughput is lower than the transmission bandwidth, a buffer underflow may increase the non-operating ratio of the bandwidth. Generally, widely used compression algorithms for an operation set or DNA compression is not designed to consider the stream transmission. As a result, the idle operating ratio of network bandwidth is higher when using a traditional compression that is based on statistics or probability focusing on arithmetic operation conducts stream transmission. This problem may be solved by the compression for reducing bottleneck of data IO as in the hardware compression of mpeg images, which can be used as an embedded method for hardware. In this study, the CUDA device was used for parallel computation. SolidStream reduces the amount of time compared to methods in which encoding and transmission are separated, as shown in Table 1.

Table 1. Times for Encoding and Transmitting the NGS Data

		Encode			Transfer Bandwidth (sec)				R_t (sec)			
		Encode time(sec)	Result Size (bytes)	Compression Ratio	1Mbit	10Mbit	100Mbit	250Mbit	1Mbit	10Mbit	100Mbit	250Mbit
SolidZipper	2C	4003.7	17,188,797,170	74.20%	137,510	13,751	1,375	550	141,514	17,755	5,379	4,554
	8C	1333.3	17,188,797,170	74.20%	137,510	13,751	1,375	550	138,844	15,084	2,708	1,883
gzip	2C	2342.4	23,388,775,996	64.90%	187,110	18,711	1,871	748	189,453	21,053	4,214	3,091
	8C	1361.7	23,388,775,996	64.90%	187,110	18,711	1,871	748	188,472	20,073	3,233	2,110
pigz	2C	1695.3	23,407,267,428	64.90%	187,258	18,726	1,873	749	188,954	20,421	3,568	2,444
	8C	811.3	23,407,267,428	64.60%	187,258	18,726	1,873	749	188,070	19,537	2,684	1,560
g-sqz	8C	4234	15,029,851,200	77.50%	120,239	12,024	1,202	481	124,473	16,258	5,436	4,715
SolidStream	2C LG		15,692,524,251	76.50%					126,859	14,236	1,588	1,384
	8C LG		15,692,524,251	76.50%					**126,855**	**13,174**	**1,352**	**500**

Note the substitution of the stream buffer schedule of the file IO (encoded NGS data block write/read). For ideal situations of the File IO ratio (NGS data block Read), when the encoding algorithm applications and data transfer (encoded NGS data block) converges to 1:1:1, the buffer is consumed so that the standby volume of the stream buffer converges to 0. However, in reality, the maximum capacity of SolidStream converges to the lowest-speed capacity of the File IO speeds of either encoding throughput or network bandwidth. The issues related to IO speed include data IO mechanisms such as SCSI, IDE, and SATA. The hardware issues related to the encoding throughput include the number of arithmetic cores participating in the operation, clock speed, or the speed of the memory

 Copyright ⓒ 2015 SERSC

bus. The network bandwidth may include network loading between the NGS platform and the analysis center.

Figure 5 illustrates R_t in seconds with respect to a change in the transfer bandwidth from 10 Mbps to 250 Mbps. In the figure, the X-axis and the Y-axis are in a logarithmic scale and in a linear scale, respectively. SolidStream outperforms the existing algorithms over a range of the transfer bandwidth. Also, the figure reveals that when data transfer speed through the internet exceeds a certain threshold, it offsets the advantages of encoding NGS data. For example, G-SQZ does not provide any advantages when the transfer speed is of 20 megabits per second. Within the current limitations of the data transfer speed, SolidStream presents the best performance among all the algorithms compared, providing a definite advantage with respect to R_t.

In order to conduct arithmetic operations for analysis in a common-use cloud service, such as Amazon Web Services (AWS), we can transmit a large amount of data in a variety of manners. First, when the analysis cluster is physically near, the data can be transmitted by a distribution service. However, when data transfer between countries is required by the cloud's operation environment, sending portable storage media to a cloud data center may be better in terms of the time saved. AWS presents a service for saving or backing up data in S3 that can be linked offline (AWS Import/ Export). The service offers data upload to users of AWS's data center, where it is stored to be subsequently imported. Before importing the saved items, it is defined by AWS onto disks and then finally sent by mailing.

It may also be possible to depend entirely on the network infrastructure. Methods of file transmission by the network include File Transfer Protocol (FTP) with a client-server model, or Peer-to-Peer (P2P) dependent on the bandwidth capacity of the transmission. As for FTP, java-based Fast Data Transfer (FDT) or Rapidant by Samsung SDS can use up to the maximum assigned network bandwidth by optimizing the protocol of the existing communication infrastructure.

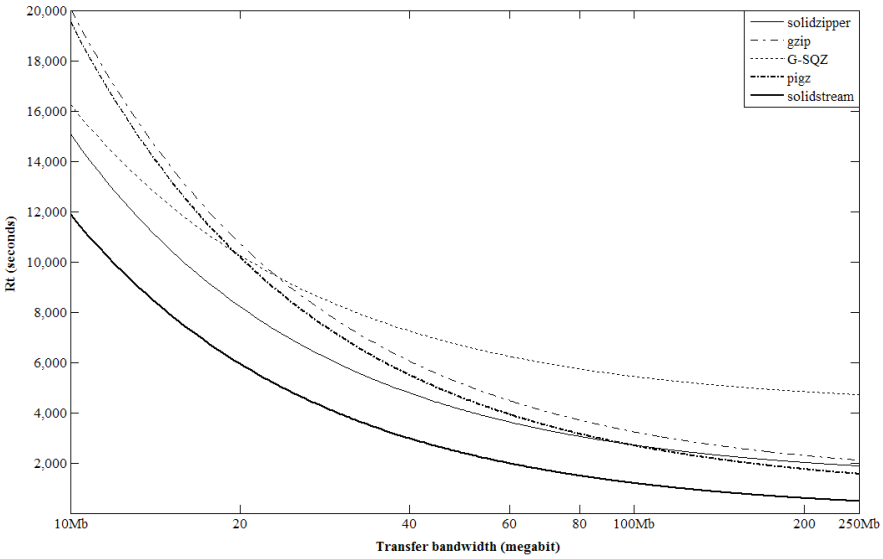

Figure 5. R_t over Various Transfer Bandwidth

This solves the problem of data transmission speed easily while still using existing softwares, and there is no need to buy additional new hardware or rent an exclusive circuit. Torrent, one of the P2P technologies, shows maximal efficiency when many users participate over a long time for dispersed distribution, but as an experimental model analyzing a large amount of DNA sequences, it may not have the efficiency required for

Copyright ⓒ 2015 SERSC

files that are distributed from limited sites to be transferred to specific Network Attached Storage (NAS), like an Amazon Simple Storage Service (Amazon S3).

Finally, streaming as one of the transmissions dependent on the network infrastructure is a method of transmission for playing sound and video multimedia. Because multimedia contents are usually much larger than text files, the cost for storing and transferring the media may be higher than those for text-based data. For reducing such cost, it is usual to store and compress the data by streaming. For the NGS data, there is a research on Large-scale DNA sequence analysis in the cloud by a stream-based approach [13], in which the NGS data produced in a sequence platform are transmitted to a cloud analysis cluster for analysis. To compensate for the differences between this paper and work by Kienzler et al. [13], this study used encoding specified for the NGS data format and imposed integrated mechanisms that managed encoding throughput and network bandwidth. However, this study did not conduct experiments with respect to the connection to analysis in the cloud of stream data.

Figure 6 illustrates R_t in seconds, when changing the transfer bandwidth from 1 Mbps to 10 Mbps. In the figure, the X-axis and the Y-axis are in logarithmic scale and in linear scale, respectively. In the figure, R_t of SolidStream is slightly lower than that of G-SQZ, rather than outperforming as shown in Figure 5. This reveals that when network bandwidth is limited within 10 Mbps, the encoding efficiency is a more important factor than encoding speed. In Figure 6, PIGZ, which is a parallel version of GZIP, shows almost the same R_t. Compared with GZIP, SolidStream and G-SQZ show R_t that is more than two times lower.

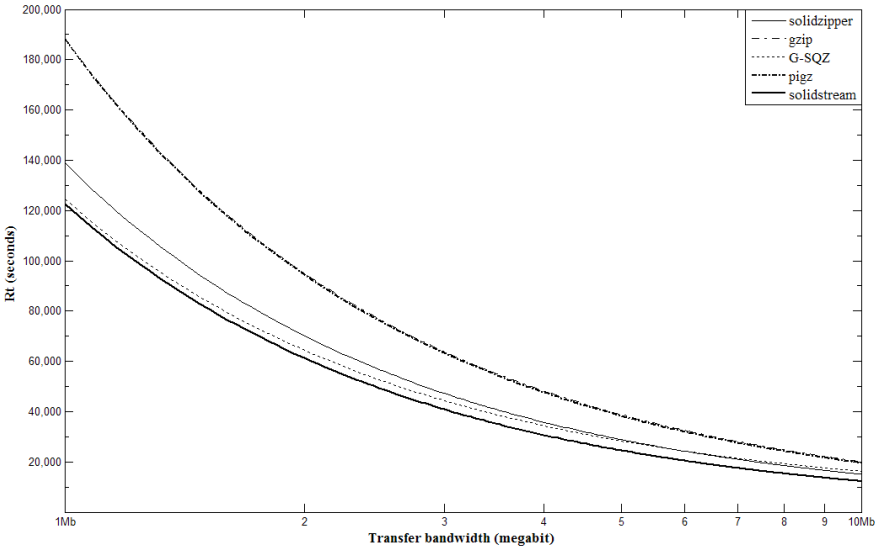

Figure 6. R_t over various Transfer Bandwidth within Range from 1 Mbps to 10 Mbps

4. Conclusion

We have addressed a method to quickly transmit a large amount of the NGS data to a cloud service in order to analyze the data efficiently using the computing power of the cloud. The proposed method, SolidStream, integrates the process of the compression and transmission into one procedure. SolidStream processes the compression and the transmission simultaneously by forwarding the NGS data in blocks to the upload stream to be sent to a cloud. Unlike existing compression algorithms that transmit data after the compression is completely finished, SolidStream reduces the entire time to transmit the compressed NGS data to the cloud and improves the efficiency of the transmission of the NGS data to the cloud.

 Copyright ⓒ 2015 SERSC

Also, SolidStream treats network transmission bandwidth to maximize data transmission efficiently to the cloud. SolidStream clearly shows benefits as a result of combining stream transfer and CUDA parallel encoding. Through the extensive experiments, we showed that SolidStream dramatically reduced the time elapsed from the beginning of the compression of the NGS data until the completion of the transmission to the cloud. SolidStream will improve the efficiency of the analysis of the NGS data.

Acknowledgements

This work was supported by the ICT R&D program of MSIP/IITP. [B0101-15-247, Development of open ICT healing platform using personal health data]

References

[1] M. Metzket, "Sequencing technologies the next generation", Nature Reviews Genetics, 11, 1 **(2010)**
[2] S. Lincoln, "The case for cloud computing in genome informatics", GenomeBiology, 11, 5 **(2010)**
[3] B. Langmead, "Searching for SNPs with cloud computing", GenomeBiology, 10, R134 **(2009)**
[4] P. Kudtarkar, "Cost-Effective Cloud Computing: A Case Study Using the Comparative Genomics Tool, Roundup", Evolutionary Bioinformatics, 6 **(2010)**
[5] Z. WeiMin, "An introduction to Tsinghua Cloud", SCIENCE CHINA Information Sciences, 53, 7 **(2010)**
[6] Y. Jeon, "SOLiDzipper: A High Speed Encoding Method for the Next-Generation Sequencing Data", Evolutionary Bioinformatics, 7 **(2011)**
[7] W. Tembe, "G-SQZ: Compact Encoding of Genomic Sequence and Quality Data", Bioinformatics, 26, 17 **(2010)**
[8] X. Chen, "DNA Compress: fast and effective DNA sequence compression", Bioinformatics, 18, 12 **(2002)**
[9] T. Soliman, "A lossless compression algorithm for DNA sequences", Int. J. Bioinform Res., **(2009)**
[10] D. Adjeroh, "DNA sequence compression using the burrows-wheeler transform", Proceedings of IEEE Computer Society Bioinformatics Conf., **(2002)** August 14-16: Palo Alto, CA, USA
[11] M. Brandon, "Data Structure and compression algorithms for genomic sequence data", Bioinformatics, **(2009)**
[12] A. CockPeter, "The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants", Nucleic Acids Research, 38, 6 **(2009)**
[13] R. Kienzler, "Largescale DNA Sequence Analysis in the Cloud: A Stream-based Approach", Proceedings of Euro-Par 6th Workshop on Virtualization in High-Performance Cloud Computing, **(2011)** August 30: Bordeaux, France
[14] Compute Unified Device Architecure (CUDA) 4.2 build, http://www.nvidia.com
[15] Apache MINA network application framework 2.0.4, http://mina.apache.org

Authors

Seokjin Im, he received the BS degree and MS degree in Electronics Engineering from Kookmin University, Korea, in 1996, 1998, respectively. He was on the research staff at Semiconductor Research Center of Hynix Semiconductor Corporation in Korea from 1999 to 2003. He received Ph. D. in Computer Science from Korea University, Korea, in 2007. From 2008 to 2010, he was a visiting scholar at California State University at Sacramento. Since 2014, he has been an associate professor of Sungkyul University in Korea. His major interests are ubiquitous computing, wireless data broadcast, spatial-temporal database, wireless network, and wireless and mobile data processing.

Copyright ⓒ 2015 SERSC

HeeJoung Hwang, he received the MS degree in Computer Science and Engineering from Inha University, Korea, in 2000. He received Ph. D. in Computer Science and Engineering from Incheon University, Korea, in 2008. From 2000, he has been in Department of Computer Engineering College of Information Technology of Gachon University as a professor. His major interests are ubiquitous computing, big data processing, software engineering, and u-Health and medical informatics.

Jinsong Ouyang, he has been in Department of Computer Science of California State University as a professor. His major interests are distributed systems especially in cloud mobile computing.

Copyright ⓒ 2015 SERSC