2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing

Scalable Assembly for Massive Genomic Graphs

Jintao Meng[1,2*], Ning Guo[1*], Jianqiu Ge[1], Yanjie Wei[1], Pavan Balaji[3], Bingqiang Wang[4]
[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, 518055, Shenzhen, China
[2] Risk Management Department, WeBank Co.，Ltd, 518057, Shenzhen, China
[3] Mathematics and Computer Science Division, Argonne National Laboratory, Lemont, IL 60439-4844, USA
[4] National Supercomputer Center in Guangzhou, Sun Yat-sen University, 510006, Guangzhou, China
E-mail: yj.wei@siat.ac.cn, balaji@anl.gov

***Abstract*—Scientists increasingly want to assemble large genomes, metagenomes, and large numbers of individual genomes. In order to meet the demand for processing these huge datasets, parallel genome assembly is a vital step. Among all the parallel genome assemblers, de Bruijn graph based ones are most popular. However, the size of de Bruijn graph is determined by the number of distinct kmers used in the algorithm, thus redundant kmers in the genome datasets donot contribute to the graph size. The scalability of genome assemblers is influenced directly by the distinct kmers in the dataset or de Bruijn graph size, rather than the input dataset size. In order to assembly large genomes, we have artificially created 16 datasets of 4 Terabytes in total from the human reference genome. The human reference genome is firstly mutated with a 5% mutation rate, and then subjected to a genome sequencing data simulator ART. The simulated datasets have linearly increasing number of distinct kmers as the size/number of the combined datasets increases. We then evaluate all five time-consuming steps of the SWAP-Assembler 2.0 (SWAP2) using these 16 simulated datasets. Compared with our previous experiment on 1000 human dataset with fixed de Bruijn graph size, the weak-scaling test shows that SWAP2 can scale well from 1024 cores using one dataset to 16,384 cores. The percentage of time usage for all five steps of SWAP2 is fixed, and total time usage is also constant. The result showed that the time usage of graph simplification occupied almost 75% of the total time usage, which will be subject to further optimization for future work.**

Keywords-weak-scaling; genome assembly; De Bruijn Graph; data simulation

**** contributed equally***

I. INTRODUCTION

There is a growing gap between the output of new generation of massively parallel sequencing machines and the ability to analyze the resulting data. Scientists increasingly want to assemble and analyze very large genomes [1], metagenomes [2,3], and large numbers of individual genomes for personalized healthcare [4,5,6,7]. In order to meet the demand for processing these huge datasets [8], parallel genome assembly seems promising, however the genome assembly algorithm/problem is very hard to scale for the following reasons [9,10].

First, the state-of-art parallel assembly solutions are dominantly utilizing the de Bruijn graph (DBG) strategy [11]. This strategy is a variant of traveling salesman problem or equivalent to the Euler path problem, which is a well-known NP-hard problem [12]. This DBG strategy is the fundamental guideline for modern second-generation sequence assemblers, such as Euler [13], Velvet [14], IDBA [15], SOAPdenovo [16], Ray[17], ABySS [18], HipMer[19], etc. However the graph reduction step in this strategy involving set operations are both computing expensive and memory exhausting [11]. In practice, assemblers have developed their own practical methods to deal with gaps and branches caused by uneven coverage, erroneous reads and repeats in graph reduction step, and a set of shorter genome sequences called contigs are generated instead of original references to simplify this problem and minimize the computing resource usage.

Second, sequencing machines are not accurate. About 50% to 80% of k-mers are erroneous [15,20], thus the nodes and edges in the graph may not be considered trustable depending on what error rate the user is willing to tolerate. What is more, kmers located in the low coverage gaps are hard to be distinguished from erroneous kmers.

Last, the number of nodes in the graph representing the genome information is enormous. One base pair (bp) in the sequencing data can generate a kmer (node) in the de Bruijn graph. However, the size of de Bruijn graph is determined only by the number of distinct kmers, thus redundant kmers in the datasets do not contribute to the graph size. The scalability of genome assemblers is influenced directly by the distinct kmers in the dataset or de Bruijn graph size, rather than the input dataset size. For example, 1000 human dataset [21,22] with 4 terabytes data contains about 2^{42} k-mers, only less than 3 billion nodes are used for building the de Bruijn graph. Assembling Hexaploid bread wheat (Triticum aestivum) genome can generate about 10~17 billion nodes in the de Bruijn graph depending on the algorithms used. For meta-genomes, the size of the graph during the assembly process can be orders of magnitude higher than those for human and wheat. This would introduce huge communication volume, computing load, and memory usage.

With the above challenges and high increasing rate of sequencing data at 5-fold a year [23], single server's computation power has been already exhausted, and the gap between the computation capacity and the sequencing data is

978-1-5090-6611-7/17 $31.00 © 2017 IEEE
DOI 10.1109/CCGRID.2017.122

still widening. Large scale parallel computing system is one option. Scalability, however, is a primary metric to prove its feasibility.

Previously we developed the SWAP-Assembler [10,24, 25], which can assemble the Yanhuang genome [26] in 26 minutes using 2,048 cores on TianHe 1A [27], 99 seconds using 16,384 cores on Tianhe 2 [25,28,29,30] and 64 seconds using 65,536 cores on Mira [25,31]. We further improved the SWAP-Assembler by analyzing and optimizing its most time-consuming steps—input parallelization, k-mer graph construction, and graph simplification (edge merging)—with the aim of developing a much faster assembly tool that can scale to hundreds of thousands of cores by using a largest genome assembly dataset of 4 terabytes. However the reference size of our previous dataset is limited as 3 billion, which means that the constructed de Bruijn graph has less than 3 billion nodes. Before this work, the record of largest assembly to date was kept by HipMer on assembling wheat with a reference of 16 Gbp [19,32].

In order to assembly large genomes and test SWAP2, we have artificially created 16 datasets of 4 Terabytes in total from the human reference genome. The largest combined dataset can generates about 50 billion nodes for the de Bruijn graph. This is about 16 times (50 billion nodes) larger than our previous work [25] and 3 times larger than wheat, and we conducted a weak scaling experiment on Mira using these 16 mutated human genome datasets with 50X coverage each. The result shows that SWAP2 can scale well from 1024 cores on assembling one dataset to 16,384 cores with 16 combined datasets. The percentage of time usage for all five steps of SWAP2 is fixed, and total time usage is also kept constant. SWAP2 is able to assemble a simulated genome (4T in total) with a de Bruijn graph of about 50 billion nodes in about 39 minutes. The result also discover that the time usage of graph simplification took almost 75% of the total time usage, which will be subject to further optimization for future work.

The rest of the paper is organized as follows. Section II briefly introduces the problem of genome assembly and related works. Section III presents the workflow on generating the simulated datasets. The performance evaluation for SWAP2 on Mira is presented in Section IV. We summarize the conclusion in Section V.

II. BACKGROUND

A. Problem Description

Given a biological genome sample with reference sequence $w \in N_g$, where N = {A, T, C, G}, g = |w|, a large number of short sequences called reads, $S = \{s_1, s_2, ..., s_h\}$, can be generated from the sequencing machines. Here s_i is a substring of w with some editorial errors introduced by sequencing machines, $1<i<h$. Genome assembly problem is to recover the reference sequence w with S.

Genome assembly with the de Bruijn graph (DBG) strategy is the process of reconstructing the reference genome sequence from these reads using de Bruijn graph consisted with k-mers. However this strategy is a variant of traveling salesman problem or equivalent to the Euler path problem, which is NP-hard [12]. Finding the original reference sequence from all possible Euler paths cannot be solved in polynomial time. What is more, gaps and branches caused by uneven coverage as well as erroneous reads and repeats prevent from obtaining a full length genome. In real cases, a set of shorter genome sequences called contigs are generated by merging unanimous paths instead. Our work focuses on finding a scalable solution for this general genome assembly problem [10, 24, 25].

B. Related works

Several state-of-art parallel assemblers have been proposed [2,9,17,19,32,33,34,35,36,38,39]. Most of them follow the de Bruijn graph (DBG) strategy proposed by Pevzner et al. in 2001 [11].

In ABySS [9], the parallelization is achieved by distributing kmers to multi-servers in order to build a distributed de Bruijn graph. Error removal and graph reduction are implemented over MPI communication primitives.

Ray [2], [17] is a general distributed engine proposed by Boisvert for traditional de Bruijn graphs, which extends k-mers (or seeds) into contigs with a heuristically greedy strategy by measuring the overlapping level of reads in both directions. Performance results on the Hg14 dataset, however indicate that Ray is 12 times slower than the SWAP-Assembler on 512 cores [10].

PASHA [33] focuses on parallelizing k-mer generation and distribution and DBG simplification in order to improve its efficiency with multithreads technology. However, PASHA allows only a single process for each unanimous path, thus limiting its degree of parallelism. Performance results [33] show that PASHA can scale to 16 cores on a machine with 32 cores on three different datasets.

YAGA [34,35] constructs a distributed de Bruijn graph by maintaining edge tuples in a community of servers. Reducible edges belonging to one unanimous path are grouped into one server with a list ranking algorithm [36]. These unanimous paths are reduced locally on separate servers. The recursive list ranking algorithm used in YAGA has a memory usage of $O(nlog(np))$, however, this induces large memory usage and causes low efficiency. Here n is the input data size, and p is the number of processes

HipMer [19,32] is an efficient end-to-end genome assembler by parallelizing the Meraculous code with both MPI and UPC language [43]. In their work, the optimizations include improving scalability of parallel k-mer analysis, a novel communication-avoiding parallel algorithm in the traversal of the de Bruijn graph and parallelizing the Meraculous scaffolding modules by leveraging the one-sided communication capabilities of UPC [37]. Experiments show that HipMer achieves a scalability of 15,360 cores on both human genome sequencing data (290 Gbp in fasta format) and wheat genome sequencing data (477 Gbp in fasta format).

Spaler [38] is a Spark and GraphX [39] based de novo genome assembler using de Bruijn graph. In Spaler, the authors parallelize the de novo genome assembly problem

with spark on distributed memory systems. Spaler used an efficient algorithm based on an iterative graph reduction technique in order to generate contigs from the DBG with a random merging approach. The authors also showed the effects of partitioning size on the running time and solving complex structure to increases the quality of the results. Comparing results shows that Spaler is faster than RAY, and ABySS with 256 cores, but still slower than SWAP-Assembler.

III. Data Simulation and Evaluation

In our previous experiment [25] part of the 1000 human dataset was used [40]. This dataset contains sequences randomly selected from the 1000 human genome sequences, and the size (or number of distinct kmers) of de Bruijn graph constructed from this dataset is about 3 billion. This prevents us from conducting a weak scaling test for all five steps of SWAP2.

To conduct a weak scaling test, the size of the de Bruijn graph should increase proportionally with the dataset size. However, real dataset is hard to meet this criteria. Instead, in this paper well-designed simulated datasets were used, with the number of distinct kmers increasing proportionally with the dataset size.

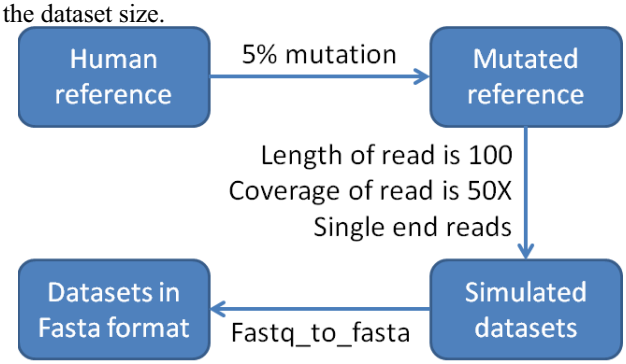

Figure 1. Workflow of data simulation

In this section, we first present the data generation workflow, and then present the evaluation of the dataset. Each of the 16 simulated datasets is firstly mutated from a given human reference genome with a 5% mutation rate, then the mutated dataset is subject to a sequencing data simulator ART [41], with the coverage parameter set as 50X. The size of each simulated data is about 250 Gbytes. The above workflow is used to generate 16 datasets, and the total size is 4 Terabytes.

A. *Workflow of data simulation*

The GRC (Genome Reference Consortium) version of Human Genome Build 37, patch release 38 is downloaded from [42] as the reference genome set. This human reference contains 1464 contigs and the total length of this reference is 3,232,546,710 bp.

The workflow of sequencing data simulation is presented in Figure 1 and described as follows:

1. **Reference mutation:** the downloaded reference is first mutated to generate a different reference dataset with a 5% random mutation rate. Each mutation is guaranteed with a unique random number within the script.
2. **Data simulation:** the simulation tool ART [41] is used to generate the sequencing data using the mutated reference dataset. In the paper we selected Illumina as the sequencing platform and set the fold of read coverage parameter as *depth*, and the length of read as *readLen* base pairs. Specially in our experiment the depth or coverage is set to be 50, and *readLen* is set to be 100.
3. **Data format conversion**: The data generated in Step 2 is in fastq format, and the fastx[43] toolset is used to convert the data into fasta format. Finally each dataset has about 250 Gbytes.

Overall 16 datasets were generated with 16 different random numbers, each with 250 Gbytes. These datasets are denoted as $S_1, S_2, \ldots S_{16}$, respectively.

B. *data analysis and evalution*

With these 16 simulated datasets, SWAP2 is used as a kmer counting tool for analysis of the simulated datasets. We run SWAP2 separately on datasets $W_1(S_1)$ of 250 Gbytes, W_2 (created by combining S_1 and S_2) of 500 Gbytes, …, W_{16} (created by combining all 16 simulation datasets) of 4 Terabytes. In these five analysis runs, the length of kmer is set as 31.

Table 1 and Figure 2 show the results on these 5 datasets $W=\{W_1, W_2, W_4, W_8, W_{16}\}$. The horizontal axis presents the corresponding combined datasets and the vertical axis presents the number of kmers with different occurring frequency in each dataset. Specially we selected the number of distinct kmers with its frequency greater or equal to 1, the number of non-unique kmers with its frequency larger than 1, and the number of kmers with its frequency larger than δ (kmers used by SWAP2 to build the de Bruijn graph), here δ is set to be 3 in our experiment. Generally, most unique kmers are generated by sequencing errors (or simulation errors in our case), and then kmers are selected by a user selected threshold δ to construct the initial de Bruijn graph in SWAP2.

Table 1. The analysis results on k-mers with different frequencies. The kmers with its frequencies larger than 3 are selected to construct the de bruijn graph. Thus the number of kmers with frequnce larger than 3 equals to the number of nodes in de Bruijn graph. (unit: billion).

Number of datasets	kmers with freq >= 1 (distinct kmer)	kmers with freq > 1 (non-unique kmer)	kmers with freq > 3 (graph size)
1	27.69	7.86	3.378
2	54.16	15.42	6.695
4	105.36	30.03	13.201
8	203.49	58.06	25.845
16	396.18	112.32	50.171

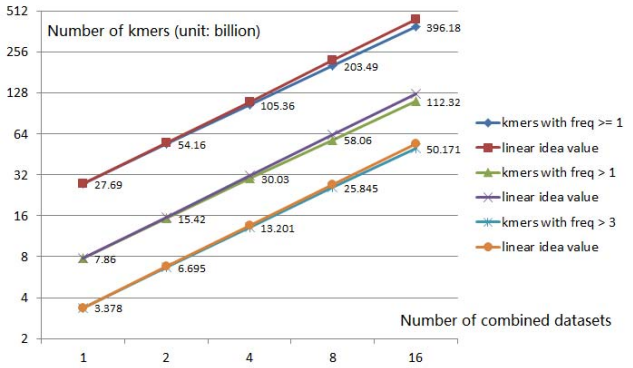

Figure 2. Kmer counting statistic results on simulated datasets. Note that the unit in this figure is billion.

From Table 1, dataset W_1 generated 3.378 billion kmers with its frequency larger than 3, while dataset W_2 produced about 6.695 billion kmers. The largest dataset W_{16} contains 50.171 billion kmers which is almost 15 times larger than W_1. Figure 2 confirms that with the increasing number of combined datasets, the number of distinct kmers, non-unique kmer and kmers with frequency larger than 3 are all growing nearly linearly. This reflects that these simulated datasets can be used in the weak-scaling experiments. In the following sections, the kmers with frequency larger than 3 were used to construct the de Bruijn graph in SWAP2 for performance evaluation.

IV. Performance Evaluation

The newly released SWAP-Assembler 2 (SWAP2 for short]) is used in this experiment; this software is available online in SourceForge [44,45]. In this performance evaluation, 32,768 computing nodes in Mira at Argonne National Laboratory [31] were allocated. Each compute node is equipped with 16 cores and 16 GB of memory; all nodes are connected with a high-speed 5D-torus network with the bidirectional bandwidth of 10 GB/s. The I/O storage system of Mira uses the IBM GPFS system; it supports parallel file I/O defined in MPI-3.

In this section, we conduct a weak scaling test with 16 simulated datasets as generated in Section III. The data size in this test was increased from one dataset (or 250 GB) to 16 datasets (or 4 TB) as the number of cores increased from 1,024 to 16,384. Here all these kmers with occurrence less than and equal to a given threshold δ are filtered, and all the remaining k-mers are used to construct the de Bruijn graph. As indicated in Table 1, the largest size is about 50 billion vertices in the initial de Bruijn graph in SWAP2. Last column in Table 1 presents the number of kmers with its frequency larger than 3 for different combined datasets. Here the filtering threshold δ is set to be 3.

Table 1 and Figure 2 show that the number of kmers with its frequency larger than 3 increases nearly linearly with the number/size of combined datasets, indicating these combined datasets can be used to conduct a weak-scaling test.

The weak-scaling test results are presented in Table 2 and Figure 3. The performance results can be analyzed from the following 3 aspects.

Scalability: Table 2 shows that SWAP2 can scale to 16,384 cores on 4 TB sequencing data. The total time usage of SWAP2 decreases slightly when the number of cores increases from 1024 to 16,384. The same trend can be found in the last four steps of SWAP2 except for the Input Parallelization step. According to Table 2 and Figure 3, one can see that the actual number of the kmers with its frequency larger than 3 in the de Bruijn graph is slightly less than the ideal value for conducting an ideal weak-scaling test. This reflects that slightly less-used kmers for large No. of cores resulted in slight decrease on the time usage in the last four steps .

Efficiency: The efficiency of SWAP2 is also affected by the number of kmers with its frequency larger than 3, and decreases slightly when the number of cores increasing from 1024 to 16,384. We re-calculated the efficiency by dividing the actual number of kmers with its frequency larger than 3 and then multiplying the ideal number of kmers, the re-calculated efficiency line (corrected line in Figure 3) is presented in figure 3. The corrected efficiency is kept to be 100% during the whole test.

Bottlenecks: The time usage of graph simplification occupied almost 75% of the total time usage, and this ratio is fixed during the whole test. Previous evaluation results in [25] with fixed de Bruijn graph size showed that the input parallelization step is the most time consuming step, however the weak scaling test in this paper has identified new bottleneck for this application, which is the subject for future work.

Although the above performance test on SWAP2 is optimized on Mira, the strategies used are general and focus only on SWAP2's major steps. The related simulation code

Table 2. Time usage of SWAP2 for the weak-scaling test. This experiment started with 1024 cores and one dataset of 50X, the dataset doubles when the number of cores doubles. The experiment scaled to 16384 cores with all 16 datasets combined. Each computing node was allocated 4 processes (ppn = 4); time is measured in seconds.

	Input Parallelization	K-mer Graph Construction	K-mer Filtering	MSG Graph Construction	Graph Simplification	Total Time usage
1024	101.92	259.94	46.12	253.68	1828.53	2490.19
2048	111.64	256.08	45.38	251.27	1800.8	2465.17
4096	112.32	253.96	44.44	247.8	1766.37	2424.89
8192	133.96	244.56	41.63	242.7	1718.89	2381.74
16384	149.17	246.71	39.01	234.78	1660.37	2330.04

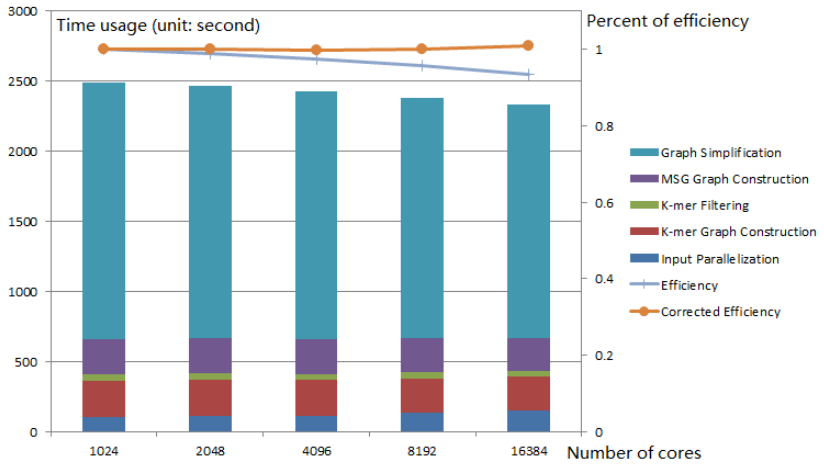

Figure 3. Accumulated time usage and efficiency of all five steps in SWAP2 for the weak-scaling test. Each computing node was allocated 4 processes (ppn = 4); time is measured in seconds.

and scripts for data simulation and performance evaluation can be downloaded from [45].

V. Conclustion

In this paper, we evaluated SWAP-Assembler 2 with a larger de Bruijn graph, 16 times larger than our previous work [26]. The previous experiment on 1000 human dataset only contain one human reference of 3 billion nucleotides. In this paper we conducted an experiment on Mira using 16 mutated human datasets with 50X coverage, and in total we simulated 4 Terabytes sequencing data with a combined reference of about 50 billion nucleotides. The result shows that SWAP2 can scale well from 1024 cores on assembling one dataset to 16,384 cores with 16 combined datasets, and SWAP2 is able to assemble a simulated genome (4T in total) with a de Bruijn graph size of about 50 billion nodes in about 39 minutes when using 16,384 cores. The percentage of time usage for all five steps of SWAP2 is fixed, and total time usage is also kept constant. The result also discover that the time usage of graph simplification occupied almost 75% of the total time usage, which will be the focus for further optimization.

Acknowledgment

This work is supported by the National Key Research and Development Program of China under Grant No. 2016YFB0201305, National High Technology Research and Development Program of China under grant No. 2015AA020109, Guangdong Provincial Department of Science and Technology under grant No. 2016B090918122, Special Program for Applied Research on Super Computation of the NSFC-Guangdong Joint Fund (the second phase), the Science Technology and Innovation Committee of Shenzhen Municipality under grant No. JCYJ20160331190123578 and JCYJ20140901003939036, Zhejiang Provincial Natural Science Foundation of China under Grant No. LQ16F020006,Youth Innovation Promotion Association, CAS to Yanjie Wei. The material is based upon work supported by the U.S. Department of Energy, Office of Science, under contract DEAC02-06CH11357. The calculations were performed on Mira at the Argonne Leadership Computing Facility.

References

[1] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G. Shan, K. Kristiansen et al., "De novo assembly of human genomes with massively parallel short read sequencing," Genome research, vol. 20, no. 2, pp. 265–272, 2010.

[2] S. Boisvert, F. Raymond, E. Godzaridis, F. Laviolette, J. Corbeil et al.,"Ray Meta: scalable de novo metagenome assembly and profiling,"Genome Biol, vol. 13, no. 12, p. R122, 2012.

[3] T. Namiki, T. Hachiya, H. Tanaka, and Y. Sakakibara, "MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads," Nucleic acids research, vol. 40, no. 20, pp. e155–e155, 2012.

[4] E. Le Chatelier, T. Nielsen, J. Qin, E. Prifti, F. Hildebrand, G. Falony, M. Almeida, M. Arumugam, J.-M. Batto, S. Kennedy et al., "Richness of human gut microbiome correlates with metabolic markers," Nature, vol. 500, no. 7464, pp. 541–546, 2013.

[5] M. Arumugam, J. Raes, E. Pelletier, D. Le Paslier, T. Yamada, D. R. Mende, G. R. Fernandes, J. Tap, T. Bruls, J.-M. Batto et al., "Enterotypes of the human gut microbiome," Nature, vol. 473, no. 7346, pp. 174–180, 2011.

[6] J. Qin, R. Li, J. Raes, M. Arumugam, K. S. Burgdorf, C. Manichanh, T. Nielsen, N. Pons, F. Levenez, T. Yamada et al., "A human gut microbial gene catalogue established by metagenomic sequencing," nature, vol. 464, no. 7285, pp. 59–65, 2010.

[7] S. R. Gill, M. Pop, R. T. DeBoy, P. B. Eckburg, P. J. Turnbaugh, B. S. Samuel, J. I. Gordon, D. A. Relman, C. M. Fraser-Liggett, and K. E. Nelson, "Metagenomic analysis of the human distal gut microbiome," science, vol. 312, no. 5778, pp. 1355–1359, 2006.

[8] J. Shendure and H. Ji, "Next-generation DNA sequencing," Nature biotechnology, vol. 26, no. 10, pp. 1135–1145, 2008.

[9] J. T. Simpson, K. Wong, S. D. Jackman, J. E. Schein, S. J. Jones, and I. Birol, "ABySS: a parallel assembler for short read sequence data," Genome research, vol. 19, no. 6, pp. 1117–1123, 2009.

[10] J. Meng, B. Wang, Y. Wei, S. Feng, and P. Balaji, "SWAP-Assembler: scalable and efficient genome assembly towards

thousands of cores," BMC bioinformatics, vol. 15, no. Suppl 9, p. S2, 2014.

[11] P. A. Pevzner, H. Tang, and M. S. Waterman, "An eulerian path approach to dna fragment assembly," Proceedings of the National Academy of Sciences, vol. 98, no. 17, pp. 9748–9753, 2001.

[12] P. Pevzner, Computational molecular biology: an algorithmic approach. MIT press, 2000.

[13] P.A. Pevzner, H. Tang, M.S. Waterman, "An Eulerian path approach to DNA fragment assembly," PNAS, vol. 98 no. 17, pp. 9748-9753, Aug, 2001.

[14] D.R. Zerbino, E. Birney, "Velvet: algorithms for de novo short read assembly using De Bruijn graphs," Genome Research, vol. 18, no.5, pp.821-829, 2008.

[15] Y. Peng, H.C.M. Leung, S.M. Yiu and F. Y. L. Chin, "IDBA - A Practical Iterative de Bruijn Graph De Novo Assembler," Proceedings of the 14th Annual international conference on Research in Computational Molecular Biology (RECOMB'10), pp. 426-440, 2010

[16] R. Li, H. Zhu, J. Ruan, W. Qian, X. Fang, Z. Shi, Y. Li, S. Li, G.Shan, K. Kristiansen, S. Li, H. Yang, J. Wang, J. Wang, "De novo assembly of human genomes with massively parallel short read sequencing," Genome Research, vol. 20, no. 2, pp. 265-272, 2010.

[17] S. Boisvert, F. Laviolette, and J. Corbeil, "Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies," Journal of Computational Biology, vol. 17, no. 11, pp. 1519–1533, 2010.

[18] J.T. Simpson, K. Wong, S.D. Jackman, J.E. Schein, S.J. Jones, I. Birol,"ABySS: a parallel assembler for short read sequence data," Genome Research, vol. 19 , no. 6 , pp. 1117-1123, 2009

[19] E. Georganas, A. Buluc¸, J. Chapman, S. Hofmeyr, C. Aluru, R. Egan, L. Oliker, D. Rokhsar, and K. Yelick, "HipMer: an extreme-scale de novo genome assembler," in Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2015, p. 14.

[20] J. Meng, J. Yuan, J. Cheng, Y. Wei, and S. Feng, "Small world asynchronous parallel model for genome assembly," in Network and Parallel Computing. Springer, 2012, pp. 145–155.

[21] N. Siva, "1000 Genomes project," Nature biotechnology, vol. 26, no. 3, pp. 256–256, 2008.

[22] Data provided by the 1000 genomes project. [Online]. Available: ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/data

[23] L.D. Stein, "The case for cloud computing in genome informatics,"Genome Biology, vol. 11, issue. 5, pp. 207, May 2010.

[24] Jintao Meng, Bingqiang Wang, Yanjie Wei, Shengzhong Feng, Pavan Balaji. SWAP-Assembler: Scalable and Efficient Genome Assembly towards Thousands of Cores, in 4th annual RECOMB satellite workshop on massively parallel sequencing (RECOMB-seq 2014), April, 2014.

[25] Jintao Meng, Sangmin Seo, Pavan Balaji, Yanjie Wei, Bingqiang Wang, Shengzhong Feng, SWAP-Assembler 2: Optimization of De Novo Genome Assembler at Extreme Scale, in Proceeding of the 45th International Conference on Parallel Processing (ICPP 2016), Philadelphia, PA, 2016

[26] G. Li, L. Ma, C. Song, Z. Yang, X. Wang, H. Huang, Y. Li, R. Li, X. Zhang, H. Yang et al., "The YH database: the first Asian diploid genome database," Nucleic acids research, vol. 37, no. suppl 1, pp. D1025–D1028, 2009.

[27] X.-J. Yang, X.-K. Liao, K. Lu, Q.-F. Hu, J.-Q. Song, and J.-S. Su,"The TianHe-1A supercomputer: its hardware and software," Journal of computer science and technology, vol. 26, no. 3, pp. 344–351, 2011.

[28] Xiangke Liao, Liquan Xiao, Canqun Yang, Yutong Lu: MilkyWay-2 supercomputer: system and application. Frontiers of Computer Science 8(3): 345-356

[29] Weixia Xu, Yutong Lu, Qiong Li, Enqiang Zhou, Zhenlong Song, Yong Dong, Wei Zhang, Dengping Wei, Xiaoming Zhang, Haitao Chen, Jianying Xing, Yuan Yuan: Hybrid hierarchy storage system in MilkyWay-2 supercomputer. Frontiers of Computer Science 8(3): 367-377

[30] Xiangke Liao, Zhengbin Pang, Kefei Wang, Yutong Lu, Min Xie, Jun Xia, Dezun Dong, Guang Suo: High Performance Interconnect Network for Tianhe System. J. Comput. Sci. Technol. 30(2): 259-272 (2015)

[31] K. Kumaran, "Introduction to Mira," in Code for Q Workshop.

[32] E. Georganas, A. Buluc¸, J. Chapman, L. Oliker, D. Rokhsar, and K. Yelick, "Parallel de Bruijn graph construction and traversal for de novo genome assembly," in International Conference for High Performance Computing, Networking, Storage and Analysis, SC14. IEEE, 2014, pp. 437–448.

[33] Y. Liu, B. Schmidt, and D. L. Maskell, "Parallelized short read assembly of large genomes using de Bruijn graphs," BMC bioinformatics, vol. 12, no. 1, p. 1, 2011.

[34] B. G. Jackson and S. Aluru, "Parallel construction of bidirected string graphs for genome assembly," in Parallel Processing, 2008. ICPP'08. 37th International Conference on. IEEE, 2008, pp. 346–353.

[35] B. G. Jackson, P. S. Schnable, and S. Aluru, "Parallel short sequence assembly of transcriptomes," BMC bioinformatics, vol. 10, no. Suppl 1, p. S14, 2009.

[36] F. Dehne and S. W. Song, "Randomized parallel list ranking for distributed memory multiprocesors," in Concurrency and Parallelism, Programming, Networking, and Security. Springer, 1996, pp. 1–10.

[37] W. W. Carlson, J. M. Draper, D. E. Culler, K. Yelick, E. Brooks, and K. Warren, Introduction to UPC and language specification. Center for Computing Sciences, Institute for Defense Analyses, 1999.

[38] A. Abu-Doleh and U. V. Catalyurek, "Spaler: Spark and graphx based de novo genome assembler," in Big Data, 2015

[39] Gonzalez J E, Xin R S, Dave A, et al. GraphX: Graph Processing in a Distributed Dataflow Framework[C], OSDI. 2014, 14: 599-613.

[40] N. Siva, "1000 Genomes project," Nature biotechnology, vol. 26, no. 3, pp. 256–256, 2008.

[41] Huang W, Li L, Myers J R, et al. ART: a next-generation sequencing read simulator[J]. Bioinformatics, 2012, 28(4): 593-594.

[42] The GRC (Genome Reference Consortium) version of Human Genome, ftp://ftp.ncbi.nih.gov/genomes/Homo_sapiens/

[43] Gordon A, Hannon G J. Fastx-toolkit[J]. FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit, 2010.

[44] SWAP-Assembler2, https://sourceforge.net/projects/swapassembler.

[45] Simulation code and scripts, https://github.com/jtmeng/SWAP-Assembler2/tree/master/utils