SWAP-Assembler 2: Scalable Genome Assembler towards Millions of Cores - Practice and Experience

Jintao Meng, Yanjie Wei*
Center for High Performance Computing
Shenzhen Institutes of Advanced Technology, CAS
Shenzhen, China
{jt.meng, yj.wei}@siat.ac.cn

Sangmin Seo, Pavan Balaji
Mathematics and Computer Science Division
Argonne National Laboratory
Chicago, USA
{sseo,balaji}@anl.gov

***Abstract*— There is widening gap between the throughput of massive parallel sequencing machines and the ability to analyze these huge sequencing data, which can be Tara bytes or even Peta bytes. Previously our assembly tool, SWAP-Assembler, can scale to 2048 cores on TianHe 1A for human Yanhuang genome. This work is to further scale SWAP-Assembler to millions of cores on Mira.**

SWAP-Assembler can be divided into 5 steps, and the most time consuming steps are input parallelization, kmer graph construction, graph simplification (edge merging). We optimize these three steps to keep the percentage of time usage in each step constant when the number of cores increases. For the input parallelization step, the input data is divided into virtual fragments with equal size, the begin position and end position for each fragment is automatically separated at the beginning symbol of reads. This data blocking strategy plays a central role in adjusting the data size to improve the communication and memory efficiency for the subsequent steps. In kmer graph construction, to prevent the communication efficiency degradation, the message size is kept constant (about 8k bytes) between any two processes by proportionally increasing the number of nucleotides to the number of processes in the input parallelization step in each round. The memory usage can be also benefited, as only a small part of the input data is processed in each round. Within graph simplification, the major improvement is to combine messages sending & receiving between its two neighbors into one loop in the communication protocol.

After integrated with the above optimizations, the new assembly tool is denoted as SWAP2 for short. In our experiment for 1k human genome dataset, the modified SWAP2 can scale to 16k cores with parallel efficiency of 70%.

Keywords: genome assembler; parallel algorithms; performance optimization

I. INTRODUCTION

There is an increasing requirement on assembling large species[1], meta-genomes[2,3], and large number of particular individuals induced by personalized healthcare [4-7]. To match this rising market on processing huge datasets generated by new generation massive parallel sequencing machines [8], parallelized genome assembly is a bottleneck and also a challenge issue in biology data processing [9,10]. Previously we have developed one assembly tool, named SWAP-Assembler [11], which can scale to 2048 cores on TianHe 1A for human Yanhuang genome. This work is to further improve its scalability and efficiency, we aim to scale SWAP-Assembler to the whole Mira (768k cores), and currently 16k cores with 70% parallel efficiency have been achieved.

The most time consuming steps in SWAP-Assembler are input parallelization, kmer graph construction, graph simplification (edge merging). We optimize these three steps to keep the percentage of time usage in each step constant when the number of cores increases. In the input parallelization step, all reads in the dataset must be move into memory. The input dataset is divided into *p* virtual fragments with almost equal size, here *p* is the number of processes. The begin position and end position for each fragment is separated at the beginning symbol of reads. Each process reads its own fragment with a given block size, the size of block can be adjusted to search the largest IO bandwidth and also make a balance between the memory usage and IO efficiency. On processing the data from 1k human genome project with 4096 nodes, results shows that on one computing node the IO bandwidth has been improved from 0.3MiB/s to 5.4 MiB/s, this is about 27% of the system theory peak performance of Mira.

In kmer graph construction, reads are cut into kmers and distributed among *p* processes. To prevent communication efficiency degradation, the message size must be constant. We proportionally increasing the number of reads with the number of processes, and automatically refill reads from disk when there is not enough reads in the memory. With this strategy both the memory usage and communication efficiency will be benefit. What's more, communication data volume has been cut by half by compressing two kmers' arc value into one.

Within graph simplification, the main improvements are to optimize the communication protocol of SWAP by removing heavy routines and minimizing the number of send & ack loops to cut down the idle communication time. For example, on processing human dataset with 32k cores, the idle time has been shrunk from 85% to 40%.

In our experiment for human dataset, the final optimized SWAP-Assembler can scale to 16k cores with an efficiency of 70.7%.

The rest of the paper is organized as follows: Section 2 presents the optimization methods for each time consumption step. The performance evaluation results for SWAP2 are given in section 3. We conclude this paper and our following work in the last section.

II. Optimization Methods

A. General Workflow of SWAP2

SWAP-Assembler has 5 steps, which include input penalization, kmer graph construction, cutoff graph, MSG graph construction and graph simplification. Figure 1 shows that the most time consuming steps are input penalization, kmer graph construction, graph simplification (edge merging). We try to optimize these three steps to keep the percentage of time usage in each step constant when the number of cores increases.

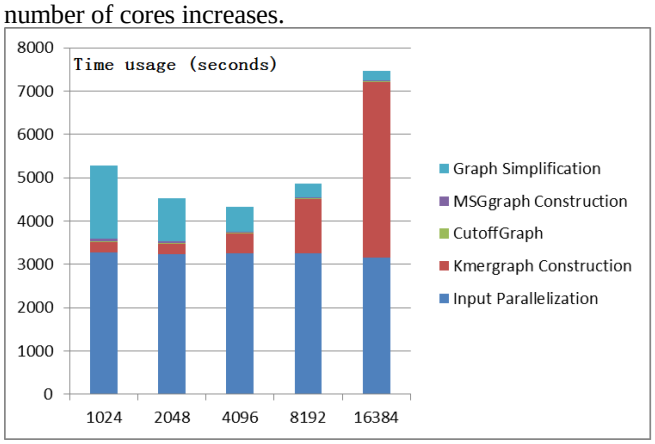

Figure 1. The statistic of the time usage for each step of SWAP-Assembler on processing the data from 1k human genome project. Here each computing node has been allocated 4 processes.

B. Input Parallelization

As the size of data generated by next generation sequencing technology generally has Tara bytes, loading these data with one process costs hours to finish [9,12]. In SWAP-Assembler, we adapt similar strategy as Ray [10] and YAGA [13], given input reads with n nucleotides from a genome of size g, we divide the input file equally into p virtual data block, p is the number of processes. Each process reads the data located in its virtual data block only once. The computational complexity of this step is bounded by O(n/p).

As the data is given in FASTA or FASTQ format, one read may be cut into two pieces and store in two nodes on two virtual data fragments. In order to overcome this shortage, SWAP2 adapts a fragment adjustment algorithm to avoid dividing the data fragment in the middle of any reads. Algorithm 1 presents the operations of this adjustment mechanism. Here each process read one data block, and search for the start point for some read. After that each process update their end point of data block by adjust their virtual fragment area. Algorithm 1 minimized the communication and I/O overhead in this adjustment phase. A positive performance improvement has been achieved in the input parallelization step. What's more, the block size for the data input on each data fragment is also set to be dynamic to maximize the IO efficiency of the input parallelization step.

Algorithm 1: Fragment adjustment
Input: dataset S in fasta or fastq format. The ID of local process *rank* and the total number of processes *p*.
Output: Virtual fragments $\{S_1, S_2, \ldots, S_p\}$.
Begin
  size = the file size of dataset S.
  step = len/p;
  start = rank*step;
  end = (rank+1)*step;
  end = end<size?end:size;

  readBuf = Read one block of data starting from *start.*
  i=0;
  While(readBuf[i]!='>' && i<blocksize) i++;
  If(rank!=0) Send i to process rank-1;
  If(rank!=p-1) receive *delta* from process rank+1;
  end += *delta*;
  S_{rank} = (start, end);
End

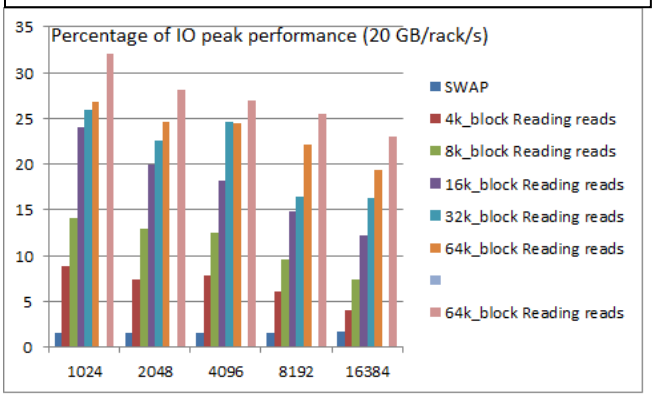

Figure 2. The IO efficiency of the input parallelization step increases with the increasing IO block size on processing the data from 1k human genome project. The last bar is the performance results with fragment adjustment algorithm.

Figure 2 shows that the IO efficiency increases steadily with the increasing block size, by removing the location part with Fragment adjustment algorithm in SWAP2, one can see a 5 percent improvement in the IO efficiency. Specially, on processing the data from 1k human genome project with 4096 cores, figure 1 shows that on one computing node the IO bandwidth has been improved from 1.5%(0.3MiB/s) to 27% (5.4 MiB/s) of the system theory peak performance of Mira.

C. *Kmer Graph Construciton*

This step aims to construct a 1-step bi-directed graph $G_k^1(s) = \{V_s, E_s^1\}$, where V_s and E_s^1 are k-molecule set and 1-step bi-directed edge set [11]. In this step, input sequences are broken into overlapping k-molecules by sliding a window of length k along the input sequence. A k-molecule can have up to eight edges, and each edge corresponds to a possible one-base extension, {A, C, G, T} in either direction. The adjacent k-molecule can be easily obtained by adding the base extension to the source k-molecule. The generated graph has O(n) *k*-molecules and O(n) bi-directed edges distributed among *p* processors. Graph construction of 1-step bi-directed graph can be achieved in O(n/p) parallel computing time, and O(n/p) parallel communication volume.

As the data size of input data has Tara bytes, the number of generated *k*-molecules is also huge for distribution. So the bottleneck of this step is communication and the communication efficiency directly affects the performance of this step. To prevent communication efficiency degradation, the message size must be constant. Then we keep he number of nucleotides processed in each communication round increase proportionally with the number of processes, more nucleotides can be automatically refilled from disk to a data pool when there is not enough nucleotides in the data pool. With this data pool, the communication part in this step and the IO part in input parallelization are isolated. We can select the best message size and IO block size for these two steps to achieve their peak performance. What's more, communication data volume and memory usage has been also cut by half by compressing two *k*-molecules' arc value into one.

Figure 3 shows that the communication bandwidth can be improved slightly by increasing the number of base pairs processed in each round. However with the increasing number of cores from 1024 to 16,384, the communication efficiency is decreasing steadily from 50% to 15%.

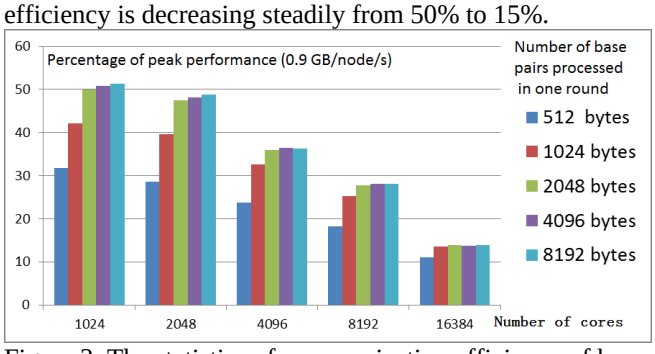

Figure 3. The statistics of communication efficiency of kmer graph construction step on processing 1k human genome dataset.

D. *Graph simplification*

SWAP computational framework with two user defined functions is used to merge edges into contigs. For each process, this step has a computing complexity of O(n/p), communication volume of nlog(log(g))/p, and communication round of O(log(log(g))).

However, according to left figure in figure 4, the idle time in the communication protocol of SWAP is increasing with the increasing number of cores. We optimize the communication protocol of SWAP by removing heavy routines and minimizing the number of send & ack loops to cut down the idle communication time. In the right figure of figure 4, with this optimization on the communication protocol, the idle time has been shrunk from 85% to 40% on processing human dataset with 32k cores.

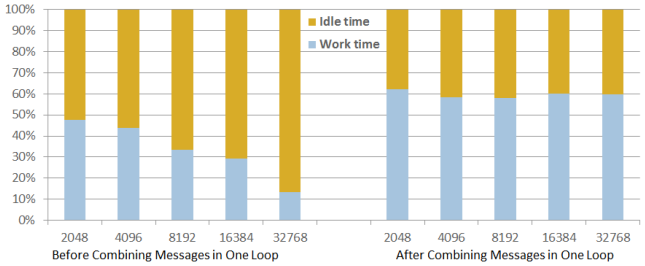

Figure 4. After the communication optimization, the idle time stops increase and keeps constant.

III. PERFORMANCE EVALUATION

The previously published software in [11] is SWAP-Assembler 0.3, SWAP2 is integrated with all the above optimizations, and this software is updated as SWAP-Assembler 0.4 in SourceForge [14]. This software is still written with C++ and MPI. But this time Mira [15] is used as our high performance cluster, and 16,384 compute nodes (262,144 cores in total) are allocated for this experiment. Each compute node is equipped with 16 cores and 16GB memory; all nodes are connected by a high-speed 5D-torus network with the bi-directional bandwidth of 10GiB/s. The I/O storage system of Mira uses IBM GPFS system; it supports Parallel File I/O defined in MPI-3. The performance and scalability of SWAP2 will be evaluated in the following subsections.

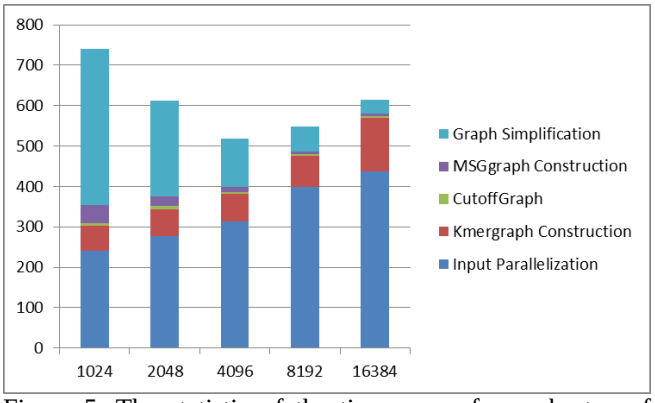

Figure 5. The statistic of the time usage for each step of SWAP2 on processing the data from 1k human genome project. Here each computing node has been allocated 4 processes.

In this experiment, the dataset is selected from 1k human genome project. In figure 1 and figure 5, one can see that overall running time of SWAP 2 is about 10X faster than SWAP. Specially, the time usage in SWAP2's input parallelization and graph simplification is about one-fifteenth and one-fifth one of its previous version. At the same time, the performance degradation in the kmer graph construction step has been resolved in SWAP2.

In order to explore the space for optimization, figure 6 illustrates the ratio of IO bandwidth, communication bandwidth, memory usage with the system resource in theory. According to this figure, one can see that the IO bandwidth of SWAP2 has been improved from 2% to 20% on 4096 cores (one rack), the communication bandwidth has been improved from 5% to 47%, and the memory usage shares the same trend with SWAP. With these results, we can conclude that SWAP2 has dramatically improved its IO bandwidth and communication bandwidth, but there is still room for further improvement. What's more the memory usage is another important point for optimization on our current work.

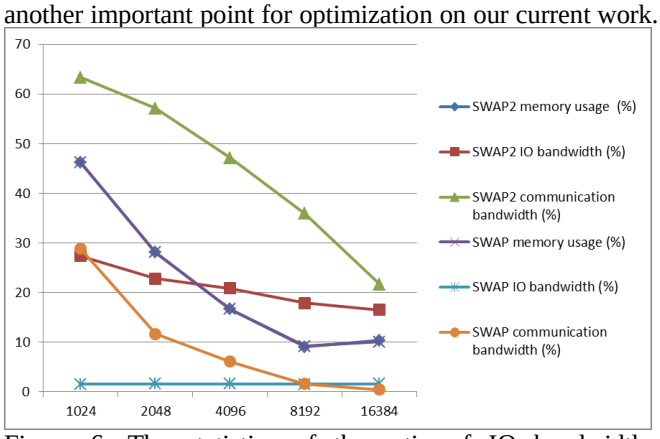

Figure 6. The statistics of the ratio of IO bandwidth, communication bandwidth, and memory usage with the system peak performance in theory. The peak performance of IO bandwidth and communication bandwidth are 20GiB/rack/s and 0.9GiB/node/s respectively. The percentage of memory usage is calculated by the memory usage of each process divided by 4 GiB.

IV. Conclusion and Future Work

In this paper, the most time consuming steps of SWAP-Assembler, including input parallelization, kmer graph construction, graph simplification, are optimized to keep the percentage of time usage in each step constant when the number of cores increases. SWAP2 is the new software intergraded with all the above optimization points. In our experiment, IO bandwidth of SWAP2 has been improved from 2% to 20% on 4096 cores (one rack), the communication bandwidth has been improved from 5% to 47%, and the memory usage shares the same trend with SWAP.

Our following work will concentrated on improving both the IO and communication efficiency. What's more, the memory usage is also an important point for optimization. Finally, we will try to improve the scalability of SWAP2 to millions of cores.

Acknowledgment

This work is supported by National Science Foundation of China under grant No. 11204342, National High Technology Research and Development Program of China under grant No. 2015AA020109, the Science Technology and Innovation Committee of Shenzhen under grant No. JCYJ20140901003939036 and KQCX20130628112914299. The work is carried out at Argonne National Lab in Chicago, and the calculations were performed on Mira.

The correspondent author of this paper may be addressed at e-mail: yj.wei@siat.ac.cn.

References

[1] Li R, Zhu H, Ruan J, Qian W, Fang X, Shi Z, Li Y, Li S, Shan G, Kristiansen K, et al: De novo assembly of human genomes with massively parallel short read sequencing. Genome research 2010, 20(2):265-272

[2] **Ray-meta:** Boisver S, Ray Meta: scalable de novo metagenome assembly and profiling, genome biology, 2012

[3] **MetaVelvet:** Namiki T, MetaVelvet: an extension of Velvet assembler to de novo metagenome assembly from short sequence reads, NAR, 2012.

[4] Le Chatelier E, Nielsen T, Qin J, et al. Richness of human gut microbiome correlates with metabolic markers[J]. Nature, 2013, 500(7464): 541-546.

[5] Arumugam M, Raes J, Pelletier E, et al. Enterotypes of the human gut microbiome[J]. Nature, 2011, 473(7346): 174-180.

[6] [16] Qin J, Li R, Raes J, et al. A human gut microbial gene catalogue established by metagenomic sequencing[J]. Nature, 2010, 464(7285): 59-65.MLA

[7] [17] Gill S R, Pop M, DeBoy R T, et al. Metagenomic analysis of the human distal gut microbiome[J]. science, 2006, 312(5778): 1355-1359.

[8] Shendure J, Ji H: Next-generation dna sequencing. Nature biotechnology 2008, 26(10):1135-1145.

[9] Simpson JT, Wong K, Jackman SD, Schein JE, Jones SJ, Birol : Abyss: a parallel assembler for short read sequence data. Genome research 2009, 19(6):1117-1123

[10] Boisvert S, Laviolette F, Corbeil J: Ray: simultaneous assembly of reads from a mix of high-throughput sequencing technologies. Journal of Computational Biology 2010, 17(11):1519-1533.

[11] SWAP-Assembler

[12] Liu Y, Schmidt B, Maskell DL: Parallelized short read assembly of large genomes using de bruijn graphs. BMC bioinformatics 2011, 12(1):354.

[13] Jackson BG, Regennitter M, Yang X, Schnable PS, Aluru S: Parallel de novo assembly of large genomes from high-throughput short reads. Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium On 2010, 1-10, IEEE

[14] SWAP-Assembler 2: http://sourceforge.net/projects/swapassembler

[15] Mira: https://www.alcf.anl.gov/user-guides/mira-cetus-vesta