SWAP-Assembler 2: Optimization of De-novo Genome Assembler at Large Scale

Jintao Meng, Sangmin Seo, Pavan Balaji, Yanjie Wei, Bingqiang Wang, Shengzhong Feng

Shenzhen Institutes of Advanced Technology, CAS
Email：{jt.meng, yj.wei}@siat.ac.cn, balaji@anl.gov

Introduction

We optimize the most time consuming steps of SWAP-Assembler, a highly-scalable genome assembler, to keep the percentage of time usage in each step constant with the increasing number of cores. The most time consuming steps including input parallelization, kmer graph construction, graph simplification (edge merging) has been optimized. The optimized assembler is denoted as SWAP-Assembler 2 (SWAP2). In our experiment using 1k human genome dataset of 4 Tara bytes, the strong scaling results shows that SWAP2 scales to 128K cores and achieved a speedup of 50X with an efficiency of 40%. Finally, the total time usage with 128K cores is about 2 minutes (including 1 minute's IO time).

Methods

Figure 1 shows the most time consuming steps in SWAP-Assembler are input penalization, kmer graph construction, graph simplification (edge merging). Specially, the dominated time consuming part locates in the input parallelization step, whose time usage is more than half of the total time usage. The time usage of kmer graph construction step increases steadily with the increasing number of cores, and the scalability of SWAP is seriously affected by this step. The graph simplification step is the third time consuming part. The time usage of this step, which should be cut by half when the number of cores doubles, decreases slightly with the number of cores.

In order to further improve both the efficiency and scalability of SWAP-Assembler, we try to optimize the above three steps by keeping the percentage of time usage in each step constant with the increasing number of cores and input data size.

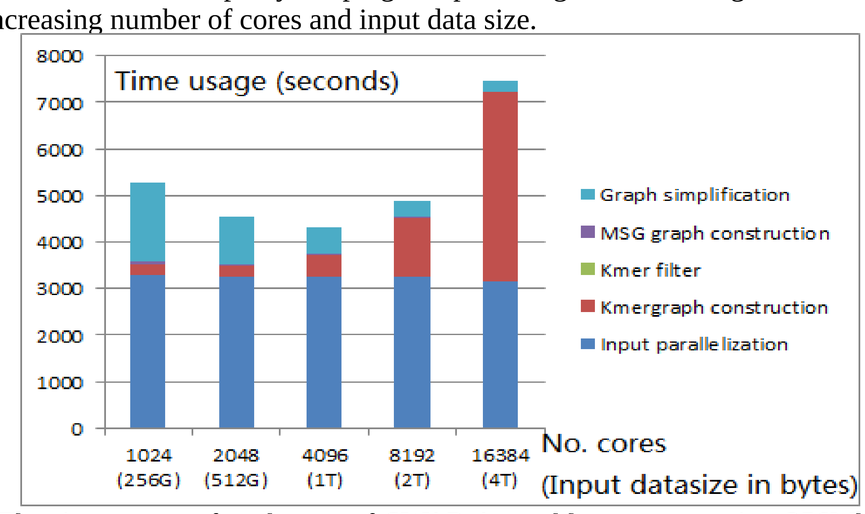

Figure 1. The time usage of each step of SWAP-Assembler on processing 256Mbytes data per core from 1k human genome project.

1. Input Parallelization

Two restrictions affect the performance of this step. One is that there is no format sensitive partition strategy for these biology data, the other is that no adjustable parameters for data block size are available to boost the IO performance as close as the system limit.

Solution:

1). Fragment adjustment algorithm (FAA) can avoid dividing data fragments in the middle of any reads. This algorithm has no I/O & communication overhead.

2). Adjusting the IO block size introduce by FAA helps improve the IO efficiency.

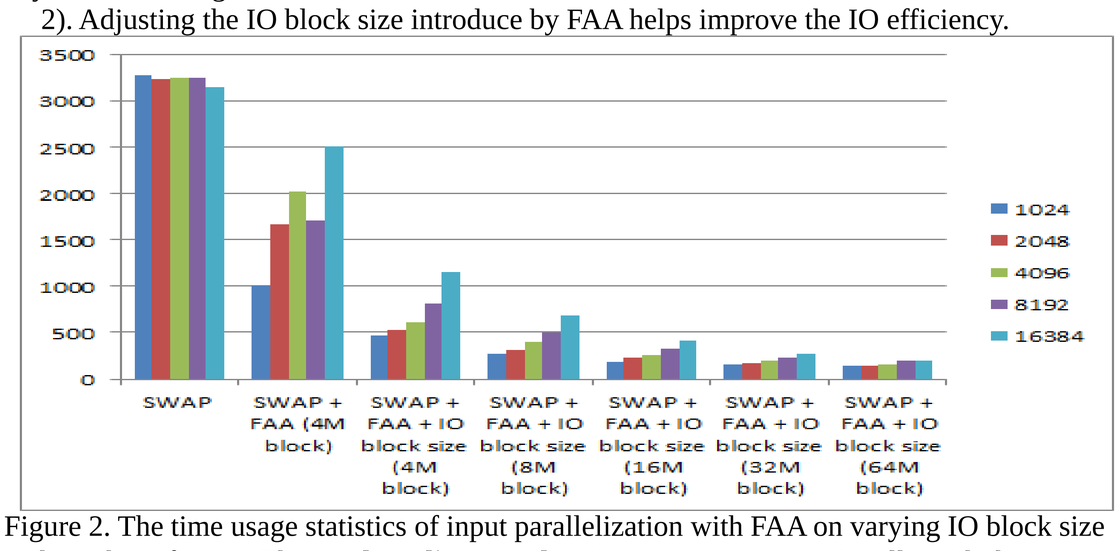

Figure 2. The time usage statistics of input parallelization with FAA on varying IO block size and number of cores. The **weak scaling test data** is increasing proportionally with the increasing number of cores and keep the problem size for each process constant on 256 MBytes.

2. **Kmer Graph Construction**

Figure 3 shows the time usage of these 3 phases on processing the data generated by 1k human genome project with SWAP and the bottleneck of this step is distribution (Communication). In SWAP-Assembler, the total data volume of messages delivered is fixed, but when the number of processes doubles, the data volume between each pair of processes will be only half and the message size will also continuously decrease proportionally. Communication with tiny messages will induce low efficiency and directly affect the performance of this step.

Solution:

1. To prevent communication efficiency degradation and boost the communication performance to its system limit, the message size is set to be constant with the increasing number of cores.

2. To resolve IO interference between input parallelization and kmer graph construction, a data buffer in figure 4 is used to insulate the IO process and communication process.

Table 1. The time usage of SWAP2 is collected for the strong scale test on human genome (Yanhuang project). (seconds)

No. cores	Input parallelization	Construct k-mer graph	Kmer filter	Construct MSG graph	Graph simplification	Total time usage
1K	117.55	281.44	19.08	185.05	1630.32	2266.72
2K	59.81	140.56	9.63	92.56	838.18	1157.72
4K	20.81	71.77	6.1	46.3	429.71	583.51
8K	13.46	39.29	3.15	22.49	223.92	307.06
16K	8.35	23.9	1.55	11.39	115.44	163.36
32K	5.08	18.99	0.88	5.71	63.77	96.51
64K	6.5	20.85	0.67	2.92	27.74	64.55

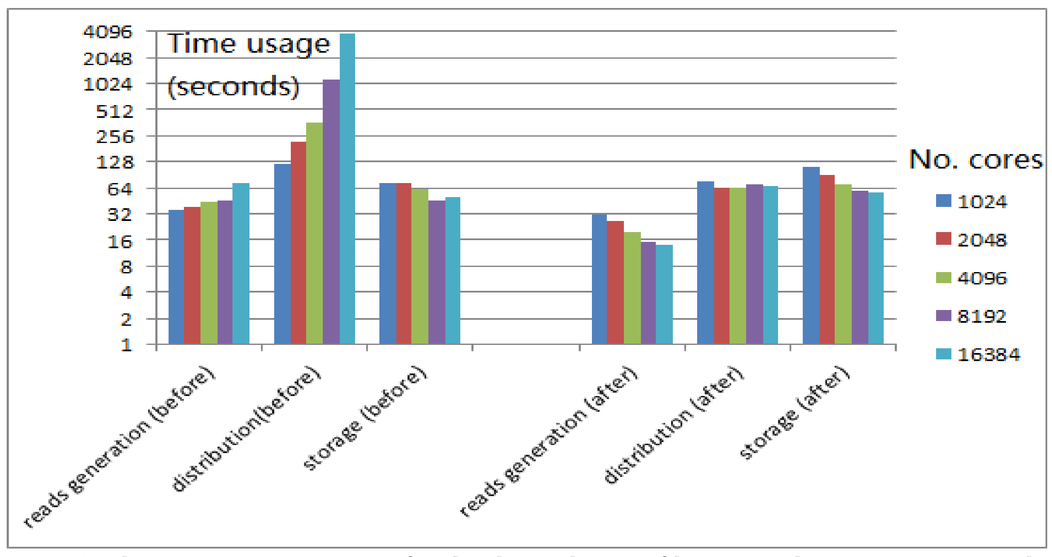

Figure 3. The time usage statistics for the three phases of kmer graph construction step before and after optimization. The running time on reads generation has decreased steadily with increasing number of cores, and 5.2X speedup has been achieved. Communication efficiency degradation in distribution step has been resolved, with 16K cores, 64X speedup has been achieved compared with our original version.

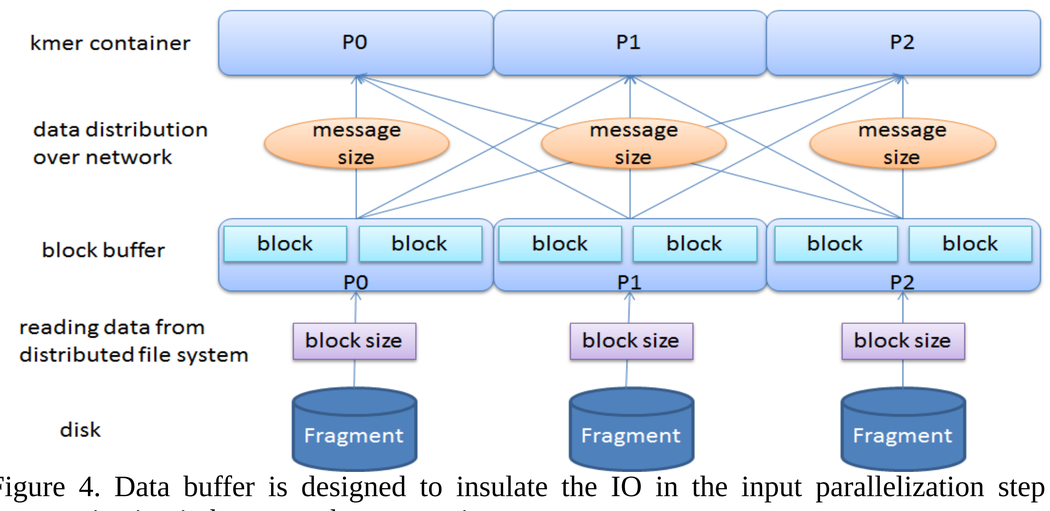

Figure 4. Data buffer is designed to insulate the IO in the input parallelization step and communication in kmer graph construction step.

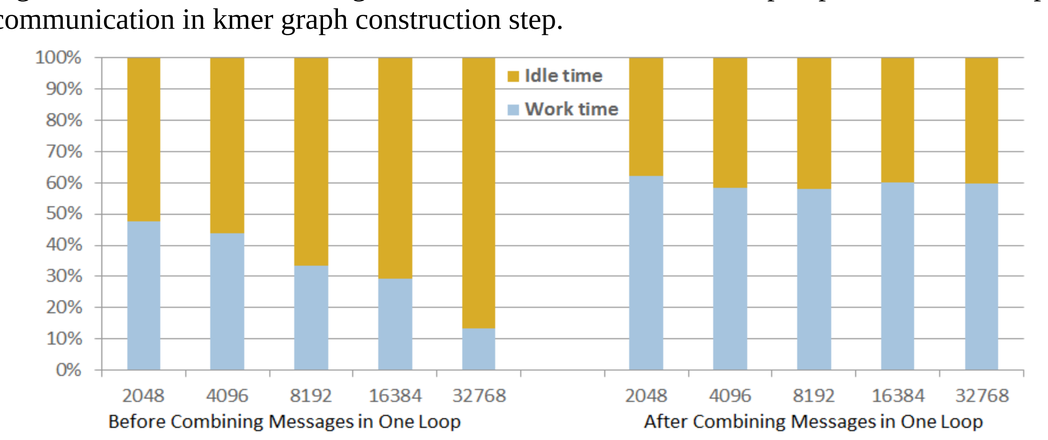

Figure 5. Percentage of time usage before and after communication protocol optimization in graph simplification.

3. Graph simplification

The idle time in the communication protocol (SWAP) of graph simplification is increasing with the increasing number of cores. After optimize the communication protocol of SWAP to minimize the number of communication loops from 4 to 2 loops and compress the idle communication time. In figure 4, the idle time has been shrunk from 85% to 40% on processing human dataset with 32k cores.

Conclusions

A strong scale experiment is to evaluate SWAP2's scalability on 4TB data selected from 1k human genome project. The time consumption results is collected and plotted in figure 6. One can see that a fixed proportion of the time usage on each step is kept as the number of cores increasing. That means all these five step are fully parallelized and scales at almost the same ratio. What's more SWAP2 achieved a speedup of 50.2X with an efficiency of 40%.

With human dataset provided by BGI's Yanhuang Project, we compared the performance of SWAP2 with other scalable assembler beyond 1024 cores, such as SWAP-Assembler. Table 1 shows that SWAP2 with 64K cores can finish this work in 64 seconds, and achieve a speedup of 35X with an efficiency of 54.8%. While with the same data SWAP-Assembler needs 26.8 minutes on 2K cores.

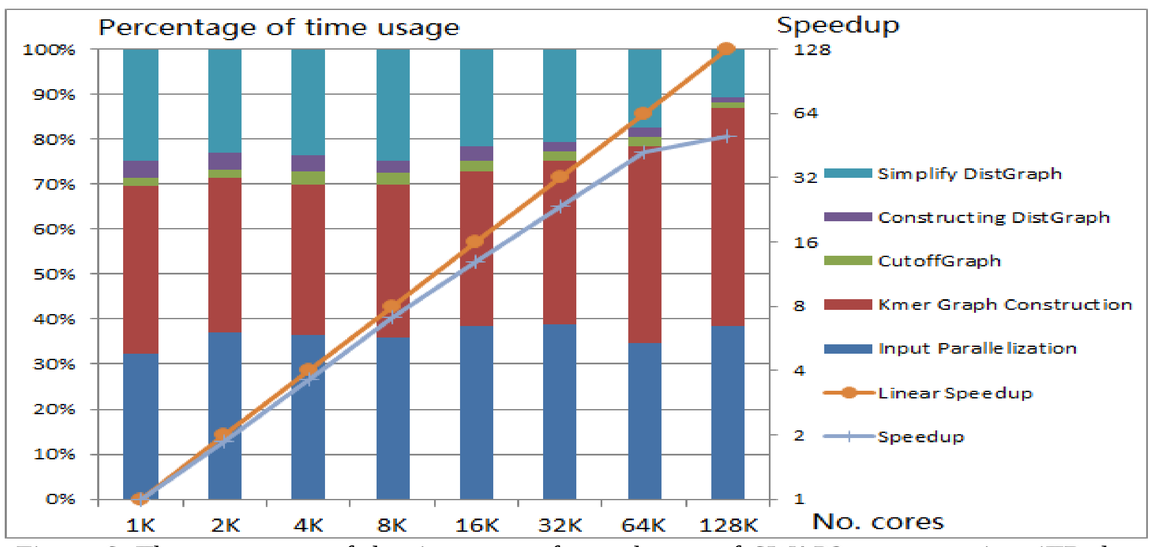

Figure 6. The percentage of the time usage for each step of SWAP2 on processing 4TB data selected from 1k human genome project.

Reference:

1. Jintao Meng, etc. SWAP-Assembler: Scalable and Efficient Genome Assembly towards Thousands of Cores, BMC Bioinformatics, Vol. 15 Supplement 9, 2014.
2. Jintao Meng, etc, SWAP-Assembler 2: Scalable Genome Assembler towards Millions of Cores - Practice and Experience, in 15th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'15)-Doctoral Symposium, May, 2015.