Bio-Medical Materials and Engineering 26 (2015) S1791–S1796
DOI 10.3233/BME-151480
IOS Press

Faster single-end alignment generation utilizing multi-thread for BWA

Heeseung Jo[a,*] and Gunhwan Koh[b]
[a]*Department of Information Technology, Chonbuk National University, Republic of Korea*
[b]*Korean Bioinformation Center, Korea Research Institute of Bioscience and Biotechnology, Republic of Korea*

Abstract. Due to next-generation sequencing (NGS) technology, genome sequencing is able to process much more data at low cost. In NGS data analysis, the mapping of sequences into a reference genome takes the largest amount of time to process. Although the Burrows-Wheeler Aligner (BWA) tool is one of the most widely used open-source software tools to align read sequences, it is still limited in that it does not fully support multi-thread mechanisms during the alignment steps. In this paper, we propose a BWA-MT tool, evolved from BWA but supporting multi-thread computation, designed to fully utilize the underlying multi-core architecture of computing resources. By using multi-thread computation, BWA-MT can significantly shorten the time needed to generate an alignment for single-end read sequences. Meanwhile, it generates an identical Sequence Alignment Map (SAM) result file as BWA. To evaluate BWA-MT, we use an evaluation system equipped with twelve cores and 32 GB memory. As a workload, we used the hg19 human genome reference sequence and various numbers of read sequences from 1M to 40M. In our evaluation, BWA-MT displays up to 3.7 times faster performance and generates an identical SAM result file to BWA. Although the increased speed might be dependent on computing resources, we confirm that BWA-MT is highly efficient and effective.

Keywords: genome sequencing, next-generation sequencing (NGS), burrow-wheeler aligner (BWA) tool, multi-thread

1. Introduction

Since the human genome project has been successfully conducted, genome sequencing technology has rapidly developed. Due to the evolution of NGS technology, it now takes less than a week to analyze the entire genome of a person. Additionally, the speed of data generation has increased with decreased cost, compared to more traditional methods. This explosive production of genome data has initiated a paradigm change in life science research.

In NGS data analysis, the mapping of read sequences into a reference genome is one of the crucial processes. This alignment analysis, however, engenders the highest cost in terms of time and computing resource consumption. To cope with NGS alignment analysis, various alignment software tools such as BWA [1], ELAND (Cox, 2007, unpublished material), MAQ [2], BOWTIE [3], and SOAP [4] have been proposed. All of them utilize indexing mechanisms to minimize execution time, and can be largely classified into hash table indexing mechanisms and suffix tree indexing

*Address for correspondence: Heeseung Jo, Department of Information Technology, Chonbuk National University, Republic of Korea. Tel.: +82-63-270-2393; Fax: +82-63-270-2394; E-mail: heeseung@jbnu.ac.kr.

mechanisms. Among them, BWA, BOWTIE, and SOAP2 are well known to achieve the highest mapping quality and performance by using the Burrows-Wheeler Transformation (BWT) algorithm [5].

Currently, BWA, an open source software, is one of the most widely used tools for whole genome alignment. The alignment process of BWA is divided into two parts: (1) mapping (BWA *aln*) and (2) alignment generation (BWA *samse/sampe*). In step (1), BWA determines suffix array (SA) coordinates for each read sequence using the pre-determined BWT index of reference genome sequences. This step of BWA allows the utilization of multiple threads to enhance performance and requires a relatively short processing time. The results are stored in the Suffix Array Index (SAI) file format. Based on the information acquired in step (1), the SAI file can be converted into actual genomic coordinates; BWA generates the output into the SAM file format through step (2). Unfortunately, because this step does not support multi-thread mechanisms, it cannot fully utilize the underlying computation resource such as multi-core processors and therefore takes substantial time to process.

Although pBWA [6] was proposed to address this issue, it is a tool based on the OpenMPI framework [7] for the parallelization of computation. It is fundamentally different from our approach in that it aims to parallelize computation to multiple computing nodes, rather than to utilize multi-core processor architecture. Moreover, since pBWA runs on the OpenMPI framework, the compilation and environment setup of pBWA can be significant obstacles for biological scientists who are not computer system engineers. Primarily, the analysis results are largely different from that of BWA in terms of mapping quality and rate.

In this paper, we design and implement a BWA-MT tool which supports multi-thread computation for the *samse* step. Thanks to multi-threading, BWA-MT can fully utilize multi-core processors and significantly shorten the necessary time to generate alignment data. Although the performance enhancement is dependent on computing resources and the number of read sequences, BWA-MT shows better performance by a magnitude of 3.7 in our evaluation and generates exactly an identical SAM result file as that generated by BWA.

2. Design and implementation

The *samse* step of BWA is reimplemented to support multi-thread computation. Our approach is to split read sequences into several subsets and process each subset on each individual thread. This approach is similar to the processing mechanism of BigData frameworks such as MapReduce [8] and Hadoop [9]. In general, they divide an entire input data stream into many segments and allocate jobs to process them using a multi-core processor and multiple nodes in parallel. Due to this parallelization mechanism, they can achieve high machine utilization and performance throughput. BWA-MT splits long read sequences into several subsets according to the number of threads set by a user. The threads then perform processing independently and concurrently.

Figures 1 and 2 depict the comparison of core alignment generation algorithms between BWA and BWA-MT. The *bwa_sai2sam_se_core* is a core function which generates alignment in BWA. First, it reads a certain amount of alignment (01-04) and converts it into sequence coordinates by labelling it the *bwa_cal_pac_pos* function (06). Then, it refines gapped alignments (07) and finally converts result into the SAM file format (09-10). These processing steps are reiterated until all input read sequences are processed. Unfortunately, this algorithm and implementation are designed to use only with a single processor core. Alternatively, in the algorithm of BWA -MT, the *bwa_sai2sam_se_core* function

creates threads named *se_worker*. According to the number of threads, the size of a read sequence subset is pre-determined, and each thread is launched with a data structure to support each read sequence subset. The *se_worker* function defines the function of a thread and performs similar computation to that of BWA.

In this processing mechanism of BWA-MT, there are two data dependency points due to the implementation of BWA. The first data dependency point is the access of the BWT index. Because the alignment generation has to reference the BWT index of reference sequences, BWA-MT must access and retain the reference information before launching each thread. Later, the BWT index stored in memory can be shared among threads. The second data dependency point is to record computation results in the SAM file format. Since this process should be ordered, BWA-MT retains each of the results in memory until all threads accomplish their function. In the final stage, BWA-MT writes them, in order, into the SAM file format.

3. Evaluation

To evaluate BWA-MT, we use an evaluation system equipped with twelve cores and 32 GB memory. The version of BWA is 0.7.5a, and BWA-MT is evolved from the same version. Table 1 shows the reference and read sequences used in our evaluations. The read sequences were retrieved from YH database [10] and NCBI [11]; we chose seven of them to vary the number of read sequences. For comparison, we run BWA and pBWA with the same option parameters. All evaluations of BWA-MT and pBWA are performed using one thread per core on twelve cores. Each evaluation result includes the execution time of the *samse* step (alignment generation).

```
00 bwa_sai2sam_se_core() {
01   while (bwa_read_seq() != 0) {
02     for (i = 0; i < number of sequences; ++i) {
03       bwa_aln2seq_core();
04     }
05
06     bwa_cal_pac_pos();
07     bwa_refine_gapped();
08
09     for (i = 0; i < number of sequences; ++i)
10       bwa_print_sam1();
11     bwa_free_read_seq();
12   }
13 }
```

Fig. 1. The core alignment generation algorithm of BWA.

```
00 bwa_sai2sam_se_core() {
01   for (j = 0; j < number of threads; ++j) {
02     if (bwa_read_seq() != 0) {
03       pthread_create(worker);
04     }
05   }
06   for (j = 0; j < number of threads; ++j)
07     pthread_join();
08
09   for (j = 0; j < number of threads; ++j) {
10     for (i = 0; i < number of sequences; ++i) {
11       bwa_print_sam1();
12     }
13   }
14 }
15
16 se_worker(void *temp) {
17   for (i = 0; i < number of sequences; ++i)
18     bwa_aln2seq_core();
19
20   bwa_cal_pac_pos();
21   bwa_refine_gapped();
22 }
```

Fig. 2. The core alignment generation algorithm of BWA-MT.

Table 1
Workloads used in evaluation

Dataset Name	Read Sequence Files	Number of Reads / Size (KB)
Ref.	hg19.fasta	6,177,547,390 (num. of char.) / 3,083,602
DS.A	070803_S85_FC5844_L8_YHCDSF.fq	1,962,893 / 238,656
DS.B	070828_S58_FC7180_L5_YHCDSD.fq	2,511,885 / 290,823
DS.C	070906_S80_FC5618_L2_YHCDSE.fq	2,833,843 / 344,872
DS.D	071005_S58_FC6492_L6_YHCDSE.fq	2,773,895 / 370,977
DS.E	071009_S80_FC5417_L8_YHCDSE.fq	4,529,006 / 596,635
DS.F	071018_S77_FC6797_L8_YHCDSE.fq	5,051,126 / 665,483
DS.G	SRR1580822.fastq	45,573,330 / 25,360,763

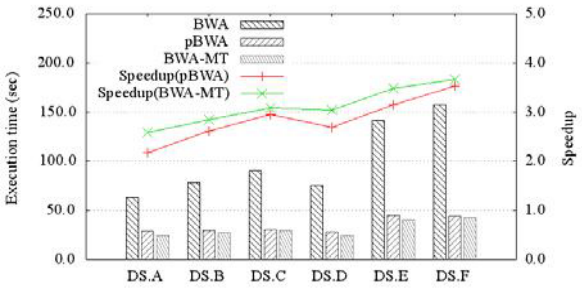

Fig. 3. The alignment generation time of BWA, pBWA, and BWA-MT for each read sequence. The lines denote the speed of pBWA and BWA-MT compared to BWA.

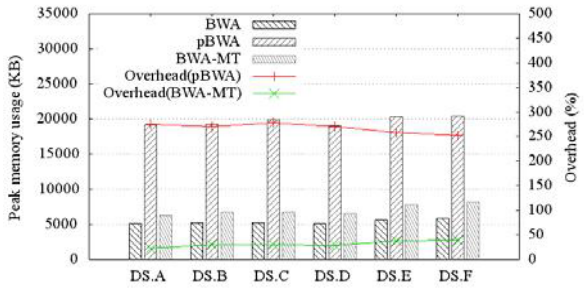

Fig. 4. The peak memory usage of BWA, pBWA, and BWA-MT for each read sequence. The lines denote the overhead percentage of pBWA and BWA-MT compared to BWA.

3.1. Execution time

Figure 3 illustrates the execution time results of BWA, pBWA, and BWA-MT for each data set. The y-axis denotes the execution time and the speed of the *samse* step. The x-axis is the read sequences utilized (also shown in Table 1); the length of each read sequence is 35 bp. As shown in Figure 3, the performance enhancement of BWA-MT surpasses that of BWA by a maximum magnitude of 3.7. As the number of read sequences increases, the performance gap also increases as the computation portion of the *samse* increase simultaneously. This shows that the multi-thread processing of BWA-MT can achieve a greater performance gap. pBWA shows a slightly longer execution time than that of BWA-MT due to the communication and barrier mechanism of the OpenMPI framework.

3.2. Memory consumption

To compare memory consumption among systems, the peak memory utilization is measured as shown in Figure 4. BWA-MT consumes more memory than BWA by 40%. This is caused by the multi-thread mechanism that retains the computation results of each thread until the completion of all

thread computations; the memory consumption is unavoidable to order to serialize the output SAM file. In our evaluation, the additional memory overhead is negligible for high performance computing at less than 3 GB. Furthermore, the memory consumption of pBWA is much larger; DS.C requires more than 19 GB of memory, which is almost three times the magnitude of memory consumed by BWA.

3.3. Scalability

Figure 5 shows the scalability of BWA-MT. This evaluation utilizes DS. G. and the read length is 100 bp. The number of reads is differentiated for each experiment and the y-axis denotes the execution time and the speed of the *samse* step. As shown in Figure 5, the execution time of BWA increases linearly as the number of read sequences increases; however, BWA-MT is less sensitive to changes in the number of read sequences. The increased speed of 40 M read sequences using BWA-MT is more than twice the magnitude of BWA, and this trend increases as the number of reads increases.

3.4. Confidence

In each workload run, the difference between the SAM results file of BWA and that of BWA-MT is compared by using *diff* command. We confirm that the two SAM results are identical, leading us to conclude that BWA-MT has the same alignment mapping quality and rate as BWA. In our evaluation, the mapping rates of datasets are 41~94%, and the percentages of multi-mappers are 0.18~0.21%. The mapping rates and the percentages of multi-mappers can differ according to datasets and BWA running options. According to this analysis, BWA-MT provides an identical SAM output to that of BWA and, consequently, produces identical confidence results.

Simultaneously, the SAM file results of pBWA are significantly different from that of BWA. Because pBWA generates multiple SAM files for a read sequence file, file outputs cannot be directly compared. For further analysis, we artificially merged the sequential results files of pBWA and discovered many mismatched alignments in the SAM result files. In the case of DS.F, the 477,056 (9.4%) alignments are mismatched.

4. Conclusion and discussion

In this paper, we propose the effectiveness of a BWA-MT tool, evolved from BWA, supporting multi-thread computation. It is designed to fully utilize the underlying multi-core architecture of computing resource. By using multi-thread computation, BWA-MT can significantly shorten the necessary time to generate alignment for single-end read sequences. Simultaneously, BWA-MT generates the same SAM result file as that generated by BWA. Confidence is significant in that most BWA users are highly sensitive about mapping quality and rate, since they provide crucial information for data analysis. We believe that BWA-MT has the potential to be highly useful to the biological scientist because of its increased speed, with the same confidence level of BWA. The only known limitation to BWA-MT is that it cannot yet process paired-end read sequences using multi-thread computation. Therefore, we plan to parallelize the *sampe* step of BWA for paired-end read sequences and to carry out further optimization in future work.

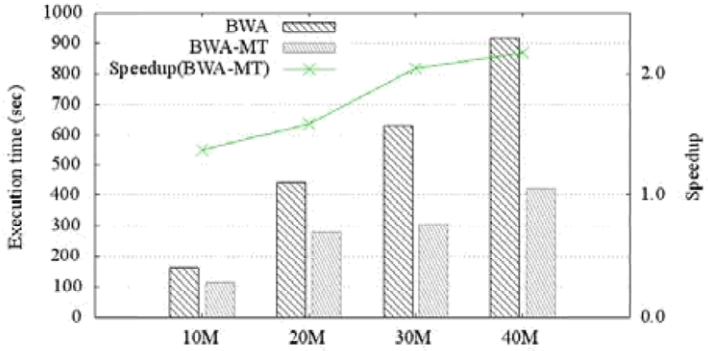

Fig. 5. The scalability of BWA-MT compared to BWA using DS.G

Acknowledgments

This work was supported by the ICT R&D program of MSIP/IITP [B0101-15-0644, The research project on High Performance and Scalable Manycore Operating System], and research funds of Chonbuk National University in 2013.

The source files will be retrieved from the source forge project page (http://sourceforge.net/projects/bwa-mt/).

References

[1] H. Li and R. Durbin, Fast and accurate short read alignment with burrows-wheeler transform, Bioinformatics **25** (2009), 1754-1760.
[2] H. Li, et al., Mapping short DNA sequencing reads and calling variants using mapping quality scores, Genome Research **18** (2008), 1851-1858.
[3] B. Langmead, et al., Searching for SNPs with cloud computing, Genome Biology **10** (2009), R134.1-R134.10.
[4] R. Li, et al., SOAP: Short oligonucleotide alignment program, Bioinformatics **24** (2008), 713-714.
[5] M. Burrows and D.J. Wheeler, A block-sorting lossless data compression algorithm, Technical Report 124, Palo Alto, CA, Digital Equipment Corporation, 1994.
[6] http://pbwa.sourceforge.net, Last visit: 7 Apr. 2015.
[7] E. Gabriel, et al., Open MPI: Goals, concept, and design of a next generation MPI implementation, Proceedings of the 11th European PVM/MPI Users' Group Meeting 2004, Budapest, Hungary, September.
[8] J. Dean and S. Ghemawat, MapReduce: Simplified data processing on large clusters, Communications of the ACM **51** (2008), 107-113.
[9] http://hadoop.apache.org (7 Apr. 2015).
[10] G. Li, et al., The YH database: The first Asian diploid genome database, Nucleic Acids Research **37** (2009), D1025-D1028.
[11] http://www.ncbi.nlm.nih.gov, Last visit: 7 Apr. 2015.