ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

Framework for Genome Assembly on GPU: A Proposal

Kajal Bangera [1], Sumashri V [1], Dr. B. Neelima [2]

Student, Department of Computer Science and Engineering, NMAM Institute of Technology, Nitte, Karkala, Karnataka India[1]

Professor, Department of Computer Science and Engineering, NMAM Institute of Technology, Nitte, Karkala, Karnataka India[2]

ABSTRACT: Genome Assembly is the process of constructing a sequence from the sample fragmented chromosomes that is the most complex computational task in the field of genomics. Further to assemble the genes there are number of assemblers available both serial as well as parallel. Parallel computing shows better performance than the serial but some parallel assemblers takes days to compute the sequence so are the serial assemblers. The framework considers computing kmers and then constructing de Bruijn graph using Graphics Processing Unit (GPU) for speeding up the k-mer and graph construction unlike the Velvet assembler. The paper gives results of k-mer slicing on Central Processing Unit (CPU). The proposed framework is under implementation and design of this new framework is promising for speeding up the genome assembly process.

KEYWORDS: Compute Unified Device Architecture (CUDA); Genome assembly; GGAKE; Graphics Processing Unit (GPU) ; Velvet;

I. INTRODUCTION

The evolution of multi-core and many-core architectures has given rise to explosion of research in terms of modifying the algorithms and designs to suite to the new and evolving architectures. In the same line, this paper also considers improving the execution time of genome assembler with the use of the latest parallel architectures. The existing assemblers are either not ported to many core architecture for example velvet [1] or ported but do not use some optimizations and not open source for example GGAKE [2]. For the same reason the paper has studied the existing assemblers and has chosen the best of few assemblers and merged into a single framework to speed up the genome assembling process.

The paper proposes a new framework that attempts to fasten the genome assembly process by parallelizing the complete methodology. It starts with dividing the input FASTA file, that contains reads to be sequenced and later extracting the reads in parallel from the sub files generated. The extracted reads are ported on to the GPU for computation of k-mers and the de Bruijn graph is constructed in parallel. After the graph is constructed it is further optimized by simplification and error removal. This approach is promising as it eliminates problems faced by few assemblers and provides a better method of computation.

This paper is organized as follow: Section II deals with background information and gives a brief summary on genome assembly. In Section III, we introduce previous tools and methods for short read assembly on different architectures. Section IV gives the design of the new idea and its implementation. Section V shows the initial results of the new design implemented. Finally, Section VI concludes the paper.

II. BACKGROUND

A cell is the building block of life. It is the basic functional, biological and structural unit of all living organism. There are many functional units within this cell; one of them is the chromosome. The chromosome is thread-like structure inside the nucleus of a cell. Each chromosome is made up of DNA and is double stranded. DNA carries the heredity

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

information from parent to off spring making each type of living creature unique. It is made of smaller units called nucleotides. There are four types of nucleotides in a DNA molecule that bonds in the following fashion: Adenine (A) bonds with Thymine (T) and Cytosine (C) bonds with Guanine (G) together forming a base pair. Base pairs are made up of the nucleotides form the DNA. This in turn forms the genes. The complete set of gene of an organism is called the genome. The genes are present in the chromosome and the chromosome is present in the nucleus of the cell.

The genome assembly refers to reconstructing a complete genome by sequencing the nucleotide bases in a DNA. Sequencing of a sample is done by breaking down the DNA sample into fragments called reads. These reads are merged and aligned to form a complete genome sequence. There are two methods that are well defined for sequencing the DNA namely Sanger Sequencing and Next Generation Sequencing (NGS). Sanger sequencing, an approach that has high read generating cost and large read size. This is used for smaller-scale projects and for validation of next-generation results for obtaining DNA reads that are greater than 500 nucleotides.

NGS is an approach where the cost of generating the reads is low and read size is small. The length of the reads varies from 35bps to a few hundred base pair. All read are read and processed by a computer program called assembler in order to assemble the reads so as the construct the genome sequence.

De novo assembly is referred to as the process of reconstructing the original genome of an organism that has neither been sequenced before nor has any available reference genome. The shotgun method is used to break the sample into fragments and a computational tool to assemble it. There are three approaches that are used by de novo assembly:

- Greedy approach
- Overlap Layout Consensus (OLC)
- De Bruijn Graph based

The greedy approach seeks the maximum overlap between the reads by comparing each and every read. The following assemblers uses this approach PCAP [3] and TIGER [4] whereas QSRA [5], VCAKE [6], SSAKE [7]. The Overlap Layout Consensus (OLC) constructs a graph called an overlap graph in order to represent the sequence .The graph is made up of reads as vertices and edges as the overlap between the two reads, overlap refers to significant amount of match between the two reads. Later multiple sequence alignment is done by bundling stretches of the graph into contigs. Finally a most likely nucleotide sequence is picked for each contig generated using Hamilton path. The assemblers that work on this approach are CAP3 [8], Edena [9]. The De Bruijn approach works by dividing the sequence into smaller sequence called reads and generate k-mer of size k. A directed De Bruijn graph is constructed using the k-mers and overlap of k-1. The k-mers represents the vertices and the k-1 overlap represents the edges. An edge is drawn between two vertices and this represents the overlap between the two sub strings. Once the graph has been constructed a traversal is done in order to find the genome sequence. The assemblers that work on this approach are ALLPATHS [10], SOAPdenovo [11], and ABySS [12]. The Hybrid methods are combination of the above three standard approach. Previously implemented assembly computation approaches for the past years were done on the CPU or cluster of CPUs. As the new technologies rise computation is done on new platforms like GPUs and its clusters.

III. Related Work

This section gives a brief summary few work carried out by previous researchers in the field genome assembly.

A. Velvet

Velvet has been implemented in c and tested on a 64 bit Linux machine that is applied to very short reads and paired ends. Its result was compared with SSAKE and VCAKE. The process of building a genome sequence using the velvet assembler consists of two parts. First is the computation of k-mer that is stored in a hash table, the data set stored in the hash table contains information about the reads and works using the command Velveth. Hash table generated is the input for constructing the de Bruijn graph. The graph construction runs using the command Velvetg as shown in figure 1.The velvet de novo assembler builds a sequence of the genome by a traversal done to the De Bruijn graph constructed. Every k-mer obtained forms a unique node of the graph as all the duplicate reads and k-mers are removed before the construction of the graph, further optimizations are done on to the graph. Optimizations include:

- **Simplification:** This is done in order to save memory cost and is done by merging the nodes that do not affect the path. The nodes to be merged should have only one path if it has multiple paths then merging the node will end up in creating wrong sequence.
- **Error removal:** The sequencing process or the sequence sample causes error in the graph. After simplification,

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

errors are removed and once the graph is error free simplification is again done. This assembler recognizes three kinds of error that are listed below:

i. **Tips**: Tips are nodes that are isolated or disconnected from one end and the information stored in the node is less than 2k (k is the k-mer length) and the arc leading to the node has a low multiplicity which as a result cannot be compared to other alternative path.

ii. **Bubbles:** When two paths in the graph starts and ends at the same node, bubbles are generated. These bubbles are usually caused by the errors in the biological variant. Velvet uses Tour Bus algorithm that is similar to Dijkstra's algorithm, breadth-first algorithm that detects the best path in the graph and helps to determine which one has to be erased.

iii. **Erroneous Connections:** Connections that neither generate correct paths nor create any recognizable structure within the graph. In velvet assembler the erroneous connections are removed after the Tour Bus algorithm, removing them with a basic coverage cut-off as specified by the user.

iv.

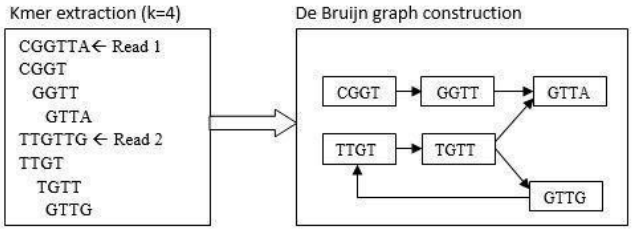

Figure1. Construction of De Brujin graph from k-mer set.

Figure1 illustrates the working of the tool "Velvet". The first box in figure 1 represents the read from the FASTA file from where the kmers are extracted. These kmers become the node to the De Bruijn graph; next step is building of edges by finding the overlap between the reads of the nodes.

B. Parallelization of velvet

Sequence assemblers are parallelized in order to gain efficiency with respect to time and space. There have been many attempts to parallelize the widely used assemblers like velvet, SOAPdenovo, YAGA [13] and ABySS. Here we focus on the parallelization of velvet .This parallel version of velvet (parallelization of velvet, "a de novo genome a sequence assembler" [14]) used hp-MPI (02.02.05.01 Linux x86-64 with three genomes Bacteria, Yeast, Worm. As mentioned earlier velvet is divided into Velveth (hashing) and Velvetg (graph). Velveth being the primer part of the assembler is been parallelized. During the parallelization of Velveth number of challenges had to be faced like, most of the processing time was spent in insertion of the k-mer to the hashing table as they were inserted sequentially. While it was parallelized it was done simultaneously but race condition occurs. There are two files in the hash table namely sequences and roadmap which is input to the Velvetg to construct the graph. This approach of parallelization includes converting the input files in parallel the converted input is stored in the sequence file. The k-mers generated inserted into the hash table. If the k-mer is already present then the reference is stored in the roadmap whereas in parallel implementation it is stored in a distributed hash table. The hash table is now distributed among the ranks so that that the data is distributed equally among the ranks .Each rank analyses its own data set and the hash-key generated depicts which part of the hash table the k-mer belongs to. Using rank address a speedup problem of communication taking longer time than computation, therefore certain number of k-mers is defined as a chunk. The process of graph construction is same as that of velvet as Velveth is untouched.

C. Dynamic Hashing Approach

De Bruijn graph based approach is very appropriate for short reads. Graph is constructed using k-mer as its vertices and the overlap as edges. The size of the graph is determined by the genome size. Dynamic hashing approach to build the De Bruijn graph for genome assembly [15] depicts an efficient way of graph construction. ABySS, Velvet and PASHA [16] are few examples of parallel de Bruijn approach. The reads here are read sequentially and repeated to construct the

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

graph. For example PASHA reads thrice, velvet reads twice and ABySS reads once (but this method gives rise to spurious edges). Further it uses binary search to search for the k-mer in the hash table. This method achieves better speed up as compared to the previous methods as this is implemented on a CPU cluster system. The design of this method is that it builds two hash vectors. The k-mer vector and property vector of same size. If the number of k- mer exceeds the size of the hash vectors, it is extended by adding same number of elements. This approach significantly has better performance than the previous approaches as it uses distributed dynamic hashing.

D. HiPGA

High Performance Genome Assembler for short Read sequence data [17] was built on a 64 CPU cores compute cluster to accelerate the de Bruijn graph based genome assembly. Here parallel input output scheme is used to make complete use of memory and computation power of the CPU. It also includes hybrid parallelism for all stages of the assembly pipeline; it shows better scalability as the number of cores in the CPU increases. It is implemented using standard C++ and MPI libraries provided by openMP and are implemented on two CPU cores of an Intel Xeon E5620 CPU. This approach is compared with ABySS, Velvet and PASHA.

E.GGAKE

GPU based Genome Assembly using K-mer extension runs on the NVidia's GPU using CUDA programming and is bench marked using 5 bacterial genomes. This produces a good quality assembly in a small amount of time. It uses character encoding for the reads in order to reduce the memory consumption. Firstly the FASTA file is read and the data is copied on to the GPU. In the GPU, computation of left and right k-mer is done by giving single read to a single thread, therefore the numbers of threads are equal to the number of reads. Now the extracted k-mers are sorted and duplicates are removed. The k-mer extension method is applied on the resultant k-mers using thrust library. Then the contig are generated and given as output. The feature that this assembler lacks is error correction. The error correction methods used by other assembler can be modified to work for GPUs to embed it into the current version of GGAKE.

IV. DESIGN AND IMPLEMENTATION

We previously discussed a few algorithms and tool that deal with genome assembly. Even though there are a huge number of serial and parallel assemblers, it still remains the most difficult computation problem in genomics. We have compared a few algorithms and have concluded with a new approach that will be implemented on the NVidia's GPU using CUDA and will be compared with the widely used assembler Velvet. The motivation behind this is to improve the performance by reducing the memory consumption and decreasing the execution time. In our approach for assembling short reads, we parallelize the input and output so that computation is done in parallel. The FASTA file is given as input to our approach. A sequence in the FASTA file format begins with a greater than symbol ">" followed by the information related to the read. The read to be extracted is located below the description. In order to compute the k-mers only reads are to be extracted. The extraction of reads becomes faster as the input FASTA file is sliced into sub files. The sub files are scanned to remove duplicate reads. Therefore the extraction of the reads is done simultaneously and the data is now sent to the GPU for further computation. Further computation includes left and right k-mer extraction, duplication removal, and construction of the directed De Bruijn graph, optimization and traversal which results to a FASTA file which consists of the final generated sequence. The design of the approach is as in the block diagram (Figure 2) given below.

Along with the extraction of k-mer the duplicates k-mers are also removed. The k-mers are added to the hash table; if the same k-mer is encountered then it is not added. Now using the k-mers all the vertices are generated and the overlaps form the edges. This is done using graph libraries in CUDA. Later the graph is optimized in parallel by simplifying the graph and removing the errors, after which a traversal is done using an efficient traversal algorithm. The sequence generated is written into a FASTA file. This approach is a combination of number of other approaches that we have discussed in Section III. The parallel computation comes from GGAKE, whereas the final division idea is from HiPGA. This idea is promising for better performance.

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

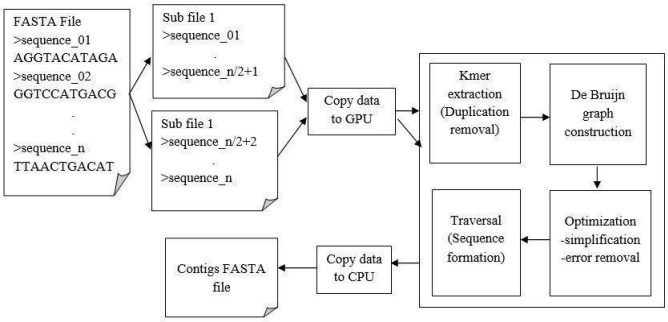

Figure 2. Design of our approach

V. RESULTS AND DISCUSSION

Based on the design above a first part that deals with division of the FASTA file and extraction of the reads in parallel is implemented on CPU. The results are as shown in Figure 3; Left k-mer, right k-mer and reverse of the right k-mer is generated. This has been now done on CPU and later will be mounted on to the GPU.As the code snippets or code to build an assembler is not freely available .Though velvet is open source it is licensed and no modifications can be done. For better understanding of the genome assembly we code it from the scratch. This design needs a few months more to be implemented completely. In this paper we briefly discuss the previous assemblers that are the motivation behind this new design.

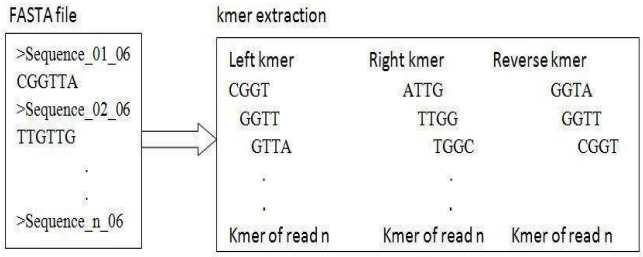

Figure 3. Initial results -extraction of reads and k-mer generation.

VI. CONCLUSION AND FUTURE WORK

In this paper we discuss approach that can be implemented on GPU for better performance. Future work includes implementing the whole design that has been discussed above. We hope our approach will perform better than the existing one because there is no end for performance improvement. The difficultly here lies in constructing graph on the GPU. Improvement of the graph construction is the scope of improvement and future work. This approach will run well for short reads it can also be modified to work well for paired or long reads. It can further be modified to reduce memory consumption by representing the reads in different forms.

REFERENCES

[1] Zerbino, Daniel R., and Ewan Birney. "Velvet: algorithms for de novo short read assembly using de Bruijn graphs." Genome research 18.5 , 821-829, 2008

[2] Anshuj Garg, Ashutosh Jain and Kolin Paul "GGAKE: GPU based Genome Assembly using K-mer Extension", High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing (HPCC_EUC), 2013 IEEE 10th International Conference, pages 1105 – 1112 , 2013

[3] Huang, Xiaoqiu, et al. "PCAP: a whole-genome assembly program." Genome research 13.9, 2164-2170,2003

ISSN(Online) : 2319-8753
ISSN (Print) : 2347-6710

International Journal of Innovative Research in Science, Engineering and Technology

(An ISO 3297: 2007 Certified Organization)

Vol. 5, Special Issue 9, May 2016

[4] Wu, Xiao-Long, et al. "TIGER: tiled iterative genome assembler." BMC bioinformatics 13.Suppl 19 : S18,2012
[5] Jr., D. W. B.; Wong, W.-K. & Mockler, T. C. , 'QSRA - a quality-value guided de novo short read assembler.', BMC Bioinformatics, volume 10, 2009
[6] Jeck, William R., et al. "Extending assembly of short DNA sequences to handle error." Bioinformatics 23.21, 2942-2944, 2007 [7] Warren, Ren L., et al. "Assembling millions of short DNA sequences using SSAKE." Bioinformatics 23.4 500-501, 2007. [8] Huang, Xiaoqiu, and Anup Madan. "CAP3: A DNA sequence assembly program." Genome research 9.9 , 868-877, 1999
[9] Hernandez, David, et al. "De novo bacterial genome sequencing: millions of very short reads assembled on a desktop computer." Genome Res. 2008 May;18(5):802-9. doi: 10.1101/gr.072033.107. Epub 2008 Mar 10.
[10] Butler J, MacCallum L, Kleber M, Shlyakhter IA, Belmonte MK, Lander ES, Nusbaum C, Jaffe DB, "ALLPATHS: De novo assembly of whole-genome shotgun microreads". Genome Research 2008, doi: 10.1101/gr.7337908. 2008 Mar 13.
[11] Li, Ruiqiang, et al. "De novo assembly of human genomes with massively parallel short read sequencing." Genome research 20.2 (2010): 265-272.
[12]Simpson, Jared T., et al. "ABySS: a parallel assembler for short read sequence data." Genome research 1117-1123. 19.6 (2009):
[13] Jackson, B. G., et al. "Parallel de novo assembly of large genomes from high-throughput short reads." Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on. IEEE, 2010.
[14]Nitin Joshi, Shashank, Shekhar srivastava, "Parallelization of Velvet,"a de novo genome sequence assembler" High Performance Computing, online link: http://hipc.org/hipc2011/studsym-papers/1569511963.pdf
[15] Kun Zhao;Weiguo Liu;Voss;Muller-Wittig W;"A Dynamic Hashing Approach to Build the de Bruijn Graph for Genome Assembly" TENCON 2013 - 2013 IEEE Region 10 Conference (31194), Pages 1 – 4 , Oct. 2013
[16] Y. Liu, B. Schmidt, and D. Maskell., "Parallelized short read assembly of large genomes using de bruijn graphs". BMC Bioinformatics, 12:354, 2011.
[17] Xiaohui Duan, Kun Zhao, Weiguo Liu "HiPGA: A High Performance Genome Assembler for Short Read Sequence Data", Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, pages 576 – 584, May 2014