**Parallelizing Long-read *de novo* Genome Assemblers On Mixed Architectures (CPU/GPU) With OpenACC and CUDA**

**By**

**Muchina Peter Kimani**

**SG30/PU/36153/19**

**A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of MASTER OF SCIENCE in Bioinformatics.**

**December,2020**

**Declaration**

I declare that this thesis is my original report that has not been presented for the award of a degree in any other university or any other award.

**NAME SIGNATURE DATE**

**MUCHINA PETER**

**SG30/PU/36153/19**

**MENTOR: SIGNATURE DATE**

**DR. JEAN-BAKA**

**Acknowledgment**

My deepest appreciation goes to God for availing the opportunity to carry out this task and His guidance through each step of this project. I am grateful to Dr Jean Baka for sparing his time for consultation and his advice towards facilitating this project.

**Abstract**

The computational challenge of de novo genome assembly (when no reference genome is available) is a hard one to tackle, when millions to billions of short reads must be assembled into contigs and further into scaffolds. Current de novo aligners (one can cite SGA, Spades, Velvet, ABySS or SOAPdenovo) try their best at finding overlaps between reads or building De Bruijn graphs on k-mers, but doing so on relatively large genomes (say, over tens or hundreds of megabases) requires both huge amounts of available RAM and CPU cycles. The very large number (in the millions or billions) of reads now routinely generated by Next-Generation Sequencing jobs (e.g. on Illumina MiSeq, HiSeq and NovaSeq) has rendered virtually intractable the OLC (overlap layout consensus) methods, which were clearly superseded by De Bruijn graph-based methods. Since the advent of GPU-based computing (where vast arrays of individually slower graphical processors are available, but with high bandwidth to main memory and high parallelism), some avenues have been explored very recently for the acceleration of De Bruijn-based (Arioc, 2018) or OLC-based GRASShopPER, 2018) assembly methods on mixed architectures (CPU + GPU). The advent of long-read sequencing technologies (Oxford Nanopore and Pacific Biosciences) brought some disruption in the landscape, and it is now possible to generate reads whose length extends over the tens or hundreds of kilobases (the current record being around 2.4MB for one read). This gives a new youth to OLC methods, and there is now the hope that such OLC methods (which were in full use at the time of the construction of the first reference human genome, based on Sanger sequencing reads) would give better accuracy than graph-based methods on long reads. The problem of their computational tractability, though, still needs to be addressed properly, and could benefit greatly from the most recent mixed architectures and software stacks enabling developers to produce accelerated code. In this project, we will explore the development of new algorithms for accelerated long-read de novo assembly on GPU architectures.

**Introduction**

We have witnessed a vital evolution in the Next generation sequencing(NGS) in the last decade. This evolution has created a platform for revealing information from unknown genomes e.g. panda. Sequencing platforms are now capable of producing billions of reads in a parallel manner. Sequencing takes lesser time and costs much less as compared to precursor platforms e.g. Sanger sequencing. The length and error rate of the reads depends on the platform used. Second generation sequencing platforms produce shorter reads with a low error rate. Third generation platforms on the other hand produce longer reads with a relatively higher error rate. Despite this advancements, reconstructing the target sequence with only the information from the reads (*de novo* assembly) is still a computational challenge. While doing de novo assembly, we make an assumption that the reads cover the target sequence and overlap each other. Using the information from the overlap we can construct the genome. However, due to technological limitations, this is barely the case. We therefore end up with contigs rather than the whole genome.

Graph construction algorithms are used in de novo assembly. They come in two flavors: de Bruijn Graph (dBG) and Overlap-Layout-Consensus (OLC). In dBG, each read is broken down to K-mers. A K-mer is a substring of the read and has a length K. A k-mer represents an edge in a directed graph. The weights of the edges represent the number of times a particular k-mer has been encountered. The next step involves finding a path that includes all the edges of the graph exactly once is found. This is known as the Eulerian path. Assemblers that use dBG include: ABySS, SOAPdenovo2 and velvet.

OLC assemblers begin by finding overlaps between the reads and finally constructing an overlap graph. In OLC, each read is a node and each edge represents the overlap between the two reads. The layout phase involves trying to find a path that corresponds to the original genome segment. An ideal graph is obtained by finding a Hamiltonian path. A path that visits each node exactly once. Assemblers that use OLC include: SGA and Dazzler.

⁠

**Statement of the Problem**

Given that the high computational capability of GPUs has not been fully utilized by existing genome assembly tools, it is crucial to develop a new GPU-based assembly framework that uses adequate techniques to scale up to the size of real-world genomes without requiring large amounts of expensive computing resources.

**Objectives**

**General objective**

To develop an application that parallelizes computer intensive step of long-read *de novo* genome assembly on GPU.

**Specific objective**

To use openACC and CUDA GPU computing models to write the application.

To compare the efficiency of openACC and CUDA models.

To compare the application with other long read de novo assemblers.

**Literature Review**

**CPU-based assembly tools**

Most current genome assemblers are CPU based and most of them use dBG in the graph construction phase. Some of the first generation assemblers include Velvet (Zerbino & Birney, 2008)⁠ and SOAPdenovo (Luo et al., 2012)⁠. Other assemblers capable of handling succinate data structures and store large graphs in memory have been proposed since the inception of first generation assemblers. This include Minia⁠ and MEGAHIT (Chikhi & Rizk, 2013)⁠ .

⁠Assemblers that have emerged to facilitate the assembly of datasets from high-throughput sequencing platforms include: Ray and SWAP (Meng et al., 2014)⁠. Among Assemblers tailored to use string graph algorithim in the graph construction phase in, SGA (Simpson & Durbin, 2012)⁠ is capable of processing large datasets in compressed formats on a single node.

**GPU-based assembly tools**

**De-Brujin graph based assemblers**

**GPU-Euler**

GPU-Euler (Mahmood, 2011)⁠ was the first GPU accerated genome assembly tool. The tool uses the Eularian path based approach. The motivation behind the tool was the advancement in multi-core technologies and the use of graphic processing units (GPU) in computer applications. GPU-Euler sough to investigate the effectiveness of graph-based sequence assembly algorithims on GPUs using CUDA programming model. Compared to loosely coupled clusters, NVIDIA GPUs programmed using CUDA model provide massive parallelism that is cheaper and easy to use.

GPU-Euler was tested on three previously assembled genomes: *Lactococcus lactis, Campilobacter jejuni* and *Neisseria meningitidis.* Reads of lengths 36, 50 and 256 base pairs (bps) were simulated for each genome. The aim of the study was to develop an assemby tool that would harness the power of GPUs effectively. The focus therefore lay in improving the run time of different parts of the algorithms more so the computer intesnsive steps. Comparing GPU-Euler with performance with previous tools, GPU-Euler showed more promise in terms of the run-time, contig accuracy and length statistics.

**String graph based assemblers**

**GAMS: Genome Assembly on Multi-GPU using String Graph**

GAMS (Jain et al., 2017)⁠ was the first GPU accelerated assembler that used the string graph-based algorithm. GAMS implements a multi-GPU based cluster.

**LaSAGNA**

LaSAGNA is GPU-Accelerated large-scale genome assembler (Goswami et al., 2018). The most computationally expensive and and memory intensive step, is the graph construction step. One limitation of the GPUs in large scale genome assembly is memory. LaSAGNA seeks to address this using a single GPU by building string graphs from approximately all-pair overlaps. It does so by using a semi-streaming approach that drastically reduces the number of disk accesses based on the available memory. To expidite the assembly pipeline, it can also run on multiple GPUs across multiple compute nodes. LaSAGNA is able to build an overlap graph from a real-world human genome dataset with a single GPU with a memory capacity of 6GB. LaSAGNA uses a two-level streaming model (i.e. from disk to host memory and host memory to device memory) that effciently utilizes the memory hierachy to minimize the disk I/O given a large-scale genome assembly task. Bench marking LaSAGNA on real-world datasets on different computing environments shows its efficiency. It can assemble a 400 GB human genome dataset in approximately 17 hours utilizing a single NVIDIA GPU (NVIDIA K40).

**GRASShopPER**

GRASShopPER (Swiercz et al., 2018)⁠, is an algorithim for de novo assembly that uses paired end reads. To add more flavour to the use of GPUs in the overlap graph construction, this tool uses a novel method for forks detection in the graph. GRASShopPER is compared to other de novo assemblers: Velvet, Celera (Istrail et al., 2004)⁠, SGA, SPAdes (Bankevich et al., 2012)⁠, Platanus (Kajitani et al., 2014)⁠ and SOAPdenovo2. Data sets used to benchmark the tool are different genomes from a simple bacterial genome to a more complex and repetitive mammalian genome. The assemblies of the data sets were evaluated with the golden standard tool QUAST. QUAST comes with a set of well-established metrics for the assembly problem. From QUAST metrics, it was observed that GRASShopPER produced contigs that covered the largest part of the genomes (metric ‘genome fraction’) and usually had a few percent more coverage than other methods. Scaffolding used an external tool and in this GRASShopPER gave better results with the SOAPdenovo2 scaffolder. For the other data sets, with less depth of coverage, GRASShopPER worked best with SSPACE scaffolder. The test environment used was Poznan Supercomputing and Networking Center on a cluster named moss. Moss provides 6 highly specialized nodes for heterogeneous computing, each equipped with two general purpose graphics processing units (GPGPUs), between 256 and 512 GB RAM, and two CPUs. Although GRASShopPER is not the most efficient among the tested assemblers with respect to memory and time usage, it represents the overlap-layout-consensus strategy, which is considered to be more accurate.

**Darwin**

Darwin (Ahmed et al., 2020)⁠ is a GPU acceleration of read overlapper for de novo assembly of long DNA reads. For simulated Pacbio reads, Darwin achieves 99.89% sensitivity and 88.30 % precision therefore highly accurate. Compared to other commonly used overlappers such as Daligner, Darwin has a higher accuracy. Darwin implements ASIC (Application-Specific Integrated Circuit). This makes it much faster than other software based overlappers. One of the downsides of ASCI however, is the bulk volume production to be economically achievable. Since using high-throughput sequencing platforms is a dynamic field, any major upgrade in the algorithm would call for a new ASIC implementation which is costly. The computational bottleneck of Darwin software has therefore been replaced with a GPU-Accelerated version. The GPU-Accelerated version is much faster than its software counterpart. This upgrade was attributed to the easy access of GPU accelerators due to their vast use. In many computing platforms, GPU accelerators have shown convincing speedups.

⁠

**Methodology**

**Workplan**

**Budget**

**References**

Ahmed, N., Qiu, T. D., Bertels, K., & Al-Ars, Z. (2020). GPU acceleration of Darwin read overlapper for de novo assembly of long DNA reads. *BMC Bioinformatics*, *21*. https://doi.org/10.1186/s12859-020-03685-1

Bankevich, A., Nurk, S., Antipov, D., Gurevich, A. A., Dvorkin, M., Kulikov, A. S., Lesin, V. M., Nikolenko, S. I., Pham, S., Prjibelski, A. D., Pyshkin, A. V., Sirotkin, A. V., Vyahhi, N., Tesler, G., Alekseyev, M. A., & Pevzner, P. A. (2012). SPAdes: A new genome assembly algorithm and its applications to single-cell sequencing. *Journal of Computational Biology*. https://doi.org/10.1089/cmb.2012.0021

Chikhi, R., & Rizk, G. (2013). Space-efficient and exact de Bruijn graph representation based on a Bloom filter. *Algorithms for Molecular Biology*. https://doi.org/10.1186/1748-7188-8-22

Goswami, S., Lee, K., Shams, S., & Park, S. J. (2018). GPU-Accelerated large-scale genome assembly. *Proceedings - 2018 IEEE 32nd International Parallel and Distributed Processing Symposium, IPDPS 2018*, 814–824. https://doi.org/10.1109/IPDPS.2018.00091

Istrail, S., Sutton, G. G., Florea, L., Halpern, A. L., Mobarry, C. M., Lippert, R., Walenz, B., Shatkay, H., Dew, I., Miller, J. R., Flanigan, M. J., Edwards, N. J., Bolanos, R., Fasulo, D., Halldorsson, B. V., Hannenhalli, S., Turner, R., Yooseph, S., Lu, F., … Venter, J. C. (2004). Whole-genome shotgun assembly and comparison of human genome assemblies. *Proceedings of the National Academy of Sciences of the United States of America*. https://doi.org/10.1073/pnas.0307971100

Jain, G., Rathore, L., & Paul, K. (2017). GAMS: Genome Assembly on Multi-GPU Using String Graph. *Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016*. https://doi.org/10.1109/HPCC-SmartCity-DSS.2016.0057

Kajitani, R., Toshimoto, K., Noguchi, H., Toyoda, A., Ogura, Y., Okuno, M., Yabana, M., Harada, M., Nagayasu, E., Maruyama, H., Kohara, Y., Fujiyama, A., Hayashi, T., & Itoh, T. (2014). Efficient de novo assembly of highly heterozygous genomes from whole-genome shotgun short reads. *Genome Research*. https://doi.org/10.1101/gr.170720.113

Luo, R., Lam, T.-W., Liu, B., Xie, Y., Li, Z., Huang, W., Yuan, J., He, G., Chen, Y., Pan, Q., Liu, Y., Tang, J., Wu, G., Zhang, H., Shi, Y., Yu, C., Wang, B., Lu, Y., Han, C., … Wang, J. (2012). SOAPdenovo2: an empirically improved memory-efficient short-read de novo assembler. *GigaScience*.

Mahmood, S. F. (2011). Department of Computer Science GPU-Euler : Sequence Assembly using GPGPU. *Analyzer*.

Meng, J., Wang, B., Wei, Y., Feng, S., & Balaji, P. (2014). SWAP-Assembler: Scalable and efficient genome assembly towards thousands of cores. *BMC Bioinformatics*. https://doi.org/10.1186/1471-2105-15-S9-S2

Simpson, J. T., & Durbin, R. (2012). Efficient de novo assembly of large genomes using compressed data structures. *Genome Research*. https://doi.org/10.1101/gr.126953.111

Swiercz, A., Frohmberg, W., Kierzynka, M., Wojciechowski, P., Zurkowski, P., Badura, J., Laskowski, A., Kasprzak, M., & Blazewicz, J. (2018). GRASShopPER—An algorithm for de novo assembly based on GPU alignments. *PLoS ONE*, *13*(8), 1–23. https://doi.org/10.1371/journal.pone.0202355

Zerbino, D. R., & Birney, E. (2008). Velvet: Algorithms for de novo short read assembly using de Bruijn graphs. *Genome Research*. https://doi.org/10.1101/gr.074492.107