Error correction in high-throughput short-read data on GPU (Literature Review)

Aman Mangal, Chirag Jain

October 22, 2014

1 Introduction

Next generation sequencing (NGS) technologies have revolutionized the way of analyzing genetic information. It has eliminated the limitations of previous techniques in terms of throughput, speed, scalability and resolution. The effectiveness of NGS technologies is further improved by using error correction methodology on the imperfect though redundant sequence reads. This literature review briefly talks about the existing sequential and parallel algorithms including GPU implementations for the error correction problem.

2 Sequential Implementations

Illumina, a popular NGS platform dominantly produces substitution errors in its output. These errors can be corrected by computing consensus bases, using highly redundant coverage information in the data. K-spectrum based approaches, first proposed by Pevzner et al.[1], generally construct k-mer[1] histogram and then try to correct reads containing low frequency k-mers. Reptile[2] uses hamming graph based approach and proposes small values of k while resolving ambiguity among possible candidates to replace a weak k-mer with solid one. BLESS[3] is highly memory efficient implementation designed for commodity computers whereas Musket[4] is an efficient multi-stage distributed corrector and suggests to correct only k-mers containing the erroneous base at either ends in a given read.

[1]substrings of length k are termed k-mers. A read of length l has $(l-k+1)$ such k-mers

Suffix tree based approach is another category of algorithms for solving this problem. Examples are SHREC[5], Hybrid SHREC[6], HiTEC[7]. Algorithms such as Coral[8] and ECHO[9] can be grouped into multiple sequence alignment based algorithms. Probabilistic approaches have also been proposed such as Quake[10], Hammer[11] etc.

3 Parallel Implementations

Length of the genome is of the order of billion base pairs in case of humans. On top of it, a big factor of renundancy is required for the error correction algorithms to precisely distinguish erroneous reads. As a result, parallel implementation becomes essential in order to quickly assemble genomes for practical purposes. Such algorithms exploit data parallelism as well as task parallelism.

Shah et al.[12] propose a spectum based parallel algorithm for distributed memory parallel computers and clusters. In this algorithm, each processor is allocated an equal share of the reads. The construction of the global spectrum is done by executing the parallel sort across all the nodes. This spectrum is then copied to the memory of each processor for the error correction of reads locally present on it. They evaluate this algorithm by plugging in their framework in the serial implementation of Reptile[2] and demonstrate a speedup of about 250x on 512 processors using large datasets.

3.1 GPU-based Implementations

In the recent years, there have been multiple attempts to solve the error correction problem using Graphics Processing Units. All the solutions being described below exploit the fact that individual reads can be corrected independently after the spectrum construction. Secondly, size of the spectrum or the k-mer frequency index built in the error correction problem can even be bigger than the GPU's global memory size. Therefore, all the GPU implementations so far have utilized the bloom filter[13], a space-efficient probabilistic data structure to hash the frequently occurring k-mers.

Shi et al.[14] proposed the first GPU based solution which could achieve 3-63 times speedup against the sequential software EULER-SR[15]. They also report that the spectrum needs to be accessed repeatedly by querying the bloom filter throughout the error correction phase which makes it their

runtime determining factor. They further improved their algorithm by incorporating the quality scores and using them to lower the computations needed for error correction[16]. Besides the better accuracy, this attempt brought them a speedup of upto 1.88x in the execution time for the error correction phase.

Liu et al.[17] implemented the first hybrid CUDA-MPI distributed version of error correction algorithm. They solve the memory constraint problem that occurs for large-scale data sets by dividing the spectrum across the nodes. However, their algorithm doesn't show runtime scalability with respect to the number of nodes used due to all-to-all reductions involved. This implementation when executed on a single node achieves up to 2.8x speedup against Shi et al.'s implementation in [14].

4 Bloom filter for GPUs

Our thorough review of the above GPU implementations leaves us with the impression that not enough attention has been paid to optimize the bloom filter's performance. Putze et al.[18] report that the naive implementation of bloom filter can be very cache-inefficient on general architectures due to random accesses required in each query. They propose some mechanisms to improve the efficiency without compromising with its accuracy. Ma et al.[19] do a detailed empirical analysis to optimize this data structure on GPUs when used for the accelerated version of sequence similarity software known as BLAST. This particular work can be useful for us as we are also solving a similar problem but for a different application.

5 Conclusion

In this review, we studied the types of algorithms and implementations to solve the error correction algorithm. We briefly discussed the sequential and parallel ways that have been worked out so far. Since our aim is to improve the performance of this algorithm on GPU, we have reviewed the related work to optimize the bloom filter data structure, an important component of all the existing GPU based implementations.

References

[1] P. A. Pevzner, H. Tang, and M. S. Waterman, "An eulerian path approach to dna fragment assembly," *Proceedings of the National Academy of Sciences*, vol. 98, no. 17, pp. 9748–9753, 2001.

[2] X. Yang, K. S. Dorman, and S. Aluru, "Reptile: representative tiling for short read error correction," *Bioinformatics*, vol. 26, no. 20, p. 2526, 2010.

[3] Y. Heo, X.-L. Wu, D. Chen, J. Ma, and W.-M. Hwu, "Bless: Bloom filter-based error correction solution for high-throughput sequencing reads," *Bioinformatics*, p. btu030, 2014.

[4] Y. Liu, J. Schrder, and B. Schmidt, "Musket: a multistage k-mer spectrum based error corrector for illumina sequence data," *Bioinformatics*, 2012.

[5] J. Schröder, H. Schröder, S. J. Puglisi, R. Sinha, and B. Schmidt, "Shrec: a short-read error correction method," *Bioinformatics*, vol. 25, no. 17, pp. 2157–2163, 2009.

[6] L. Salmela, "Correction of sequencing errors in a mixed set of reads," *Bioinformatics*, vol. 26, no. 10, pp. 1284–1290, 2010.

[7] L. Ilie, F. Fazayeli, and S. Ilie, "Hitec: accurate error correction in high-throughput sequencing data," *Bioinformatics*, vol. 27, no. 3, pp. 295–302, 2011.

[8] L. Salmela and J. Schrder, "Correcting errors in short reads by multiple alignments," *Bioinformatics*, vol. 27, no. 11, pp. 1455–1461, 2011.

[9] W.-C. Kao, A. H. Chan, and Y. S. Song, "Echo: A reference-free short-read error correction algorithm," *Genome Research*, vol. 21, no. 7, pp. 1181–1192, 2011.

[10] D. R. Kelley, M. C. Schatz, and S. L. Salzberg, "Quake: quality-aware detection and correction of sequencing errors," *Genome Biology*, vol. 11, no. 11, p. R116, 2010.

[11] P. Medvedev, E. Scott, B. Kakaradov, and P. Pevzner, "Error correction of high-throughput sequencing datasets with non-uniform coverage," *Bioinformatics*, vol. 27, no. 13, pp. i137–i141, 2011.

[12] A. R. Shah, S. Chockalingam, and S. Aluru, "A parallel algorithm for spectrum-based short read error correction," in *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*, pp. 60–70, IEEE, 2012.

[13] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[14] H. Shi, B. Schmidt, W. Liu, and W. Müller-Wittig, "A parallel algorithm for error correction in high-throughput short-read data on cuda-enabled graphics hardware," *Journal of Computational Biology*, vol. 17, no. 4, pp. 603–615, 2010.

[15] M. J. Chaisson and P. A. Pevzner, "Short read fragment assembly of bacterial genomes," *Genome research*, vol. 18, no. 2, pp. 324–330, 2008.

[16] H. Shi, B. Schmidt, W. Liu, and W. Müller-Wittig, "Quality-score guided error correction for short-read sequencing data using cuda," *Procedia Computer Science*, vol. 1, no. 1, pp. 1129–1138, 2010.

[17] Y. Liu, B. Schmidt, and D. L. Maskell, "Decgpu: distributed error correction on massively parallel graphics processing units using cuda and mpi," *BMC bioinformatics*, vol. 12, no. 1, p. 85, 2011.

[18] F. Putze, P. Sanders, and J. Singler, "Cache-, hash-and space-efficient bloom filters," in *Experimental Algorithms*, pp. 108–121, Springer, 2007.

[19] L. Ma, R. D. Chamberlain, J. D. Buhler, and M. A. Franklin, "Bloom filter performance on graphics engines," in *Parallel Processing (ICPP), 2011 International Conference on*, pp. 522–531, IEEE, 2011.