engineering proceedings

Proceeding Paper

Performance Optimization of a Parallel Error Correction Tool †

Marco Martínez-Sánchez *, Roberto R. Expósito and Juan Touriño

Computer Architecture Group, CITIC, Universidade da Coruña, 15071 A Coruña, Spain; rreye@udc.es (R.R.E.); juan@udc.es (J.T.)
* Correspondence: marco.msanchez@udc.es
† Presented at the 4th XoveTIC Conference, A Coruña, Spain, 7–8 October 2021.

Abstract: Due to the continuous development in the field of Next Generation Sequencing (NGS) technologies that have allowed researchers to take advantage of greater genetic samples in less time, it is a matter of relevance to improve the existing algorithms aimed at the enhancement of the quality of those generated reads. In this work, we present a Big Data tool implemented upon the open-source Apache Spark framework that is able to execute validated error-correction algorithms at an improved performance. The experimental evaluation conducted on a multi-core cluster has shown significant improvements in execution times, providing a maximum speedup of 9.5 over existing error correction tools when processing an NGS dataset with 25 million reads.

Keywords: high performance computing; Big Data; bioinformatics; Next Generation Sequencing

check for updates

Citation: Martínez-Sánchez, M.; Expósito, R.R.; Touriño, J. Performance Optimization of a Parallel Error Correction Tool. *Eng. Proc.* **2021**, *7*, 34. https://doi.org/10.3390/engproc2021007034

Academic Editors: Joaquim de Moura, Marco A. González, Javier Pereira and Manuel G. Penedo

Published: 15 October 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, the development of effective and fast techniques for processing large volumes of genetic data has gained relevance due to the need of counting on these reads for the evolution of biology-related scientific fields. As a direct consequence of this necessity, new approaches to solve this problem have been presented throughout the last decade under the name of Next Generation Sequencing (NGS) [1]. However, even though these technologies are able to read amounts of genetic samples that are orders of magnitude larger than the previous alternatives [2], they still do not have an insignificant error rate for their generated reads. Hence, multiple algorithms have been proposed in the literature to correct these mistakes in the samples and make up higher quality reads. Among them, CloudEC [3] is a Big Data tool built upon the Apache Hadoop framework [4] that is able to perform corrections to genetic datasets by running multiple steps of alignments of the input samples, and replacing the bases with the lowest qualities of all those aligned samples with another representations of higher quality.

At the same time, significant progress has also been made in the Big Data field, where for many years some of the main approaches were based on the MapReduce paradigm [5], a programming model proposed by Google that defines multiple programmable and non-programmable phases to decouple the data transformation logic from the communication and load distribution tasks. However, some alternatives have also been proposed with this goal in mind, as it is with the Apache Spark framework [6], that are able to relieve both the data scientists and Big Data developers from directly operating with the MapReduce framework and allow them to tackle with a higher-level API. Moreover, this programming interface also comes with some optimizations not usually found in another technologies, such as the lazy computational model and the usage of the main memory for storing the data instead of secondary storage.

In this context, it makes sense to develop a new tool aimed at solving the computational challenges introduced when correcting large NGS datasets, taking advantage of Apache Spark to improve the performance of existing error correction algorithms.

2. Development

With this work, we are proposing a parallel tool able to process input FASTQ reads, an extended unaligned sequence format, also supporting the specific format internally used by CloudEC. The overall workflow of the tool can be decomposed in the six phases shown in Figure 1, which basically consist of: two mandatory stages at the beginning and the end of the pipeline to preprocess the input and format the output, respectively; two error correction phases that implement different algorithms each one; and, finally, two filtering stages that are able to tag the input reads with auxiliary information for the error correction phases. This processing pipeline is also used in the original CloudEC tool, which has served as the underlying baseline for our Spark-based implementation.

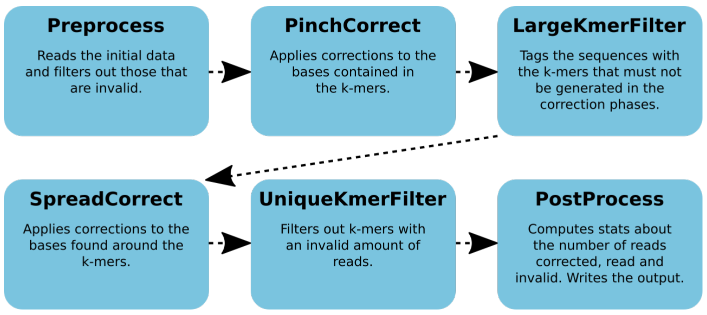
Preprocess
Reads the initial data and filters out those that are invalid.
PinchCorrect
Applies corrections to the bases contained in the k-mers.
LargeKmerFilter
Tags the sequences with the k-mers that must not be generated in the correction phases.
SpreadCorrect
Applies corrections to the bases found around the k-mers.
UniqueKmerFilter
Filters out k-mers with an invalid amount of reads.
PostProcess
Computes stats about the number of reads corrected, read and invalid. Writes the output.

Figure 1. The six phases of the error correction tool.

For the development of our proposal, the baseline tool have been redesigned and reimplemented from scratch by replacing the Hadoop framework with Apache Spark. Since the error correction algorithms implemented by CloudEC are based on the alignment of nucleotide subsequences from the input reads, it is expected to significantly improve performance given the ability of Spark to store this temporary alignment data in the main memory instead of on a disk.

3. Experimental Evaluation

To determine the performance improvements provided by our proposal over the original CloudEC tool, the experimental evaluation has been carried out on the Pluton cluster of the Computer Architecture Group. In particular, the cluster nodes used in the experiments consist of two Intel Xeon E5-2660 octa-core processors at 2.2 GHz (i.e., 16 cores per node), 64 GiB of main memory, and one 1 TiB local SATA disk intended for intermediate data storage during the executions.

In order to provide reproducible results, the datasets used in the experiments have been obtained from SRA [7], an open repository for genomic reads. In particular, this evaluation has been conducted using two public datasets in FASTQ format whose specific features are shown in Table 1.

Table 1. Tag, SRA accessor identifier and features of the public datasets used in the experiments.

Tag	Identifier	Number of Sequences	Sequence Length
D1	SRR034509	10,353,618	202 bp
D2	SRR4291508	25,232,347	100 bp

Regarding software settings, the evaluation was undertaken with Spark 2.3.4 and Hadoop 2.9.2. Additionally, all the experiments were executed using OpenJDK 1.8.0_242. Finally, the length of the k-mers (i.e., the length of the subsequences that were used for the alignment tasks) has been set to 24 (k = 24) for both tools to ensure a fair comparison.

4. Results and Conclusions

Table 2 provides the execution times of our tool and the speedups over CloudEC for both datasets when using different number of nodes. These results show that our implementation significantly outperforms CloudEC for all the scenarios under evaluation. The maximum speedups are obtained when using 13 nodes: 2.8× and 9.5× for D1 and D2 datasets, respectively. Note that the speedups obtained are clearly higher for D2, which contains twice as many reads as D1 (see Table 1), showing the benefits of our tool when processing more compute-intensive datasets. Moreover, the scalability is improved when compared to CloudEC, as can be noted by the fact that the speedups increase monotonically with the number of nodes.

Table 2. Results of the performance evaluation of our tool. The execution times are shown in seconds for each dataset and number of cluster nodes. The speedup provided by our tool over CloudEC is shown in parentheses.

Dataset	5 Nodes	9 Nodes	13 Nodes
D1	4865 *(2.4×)*	1885 *(2.6×)*	1113 *(2.8×)*
D2	7473 *(5.8×)*	2484 *(8.2×)*	1511 *(9.5×)*

Overall, these results are very encouraging, proving that the Spark-based implementation is able to handle realistically sized NGS datasets, providing results in bounded times that are even lower than the execution times of another tools of the state of the art.

Author Contributions: Conceptualization, R.R.E. and J.T.; methodology, M.M.-S., R.R.E. and J.T.; implementation, M.M.-S.; validation, M.M.-S.; writing—original draft preparation, M.M.-S.; writing—review and editing, R.R.E. and J.T. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Innovation of Spain (PID2019-104184RB-I00/AEI/10.13039/501100011033), and by Xunta de Galicia and FEDER funds of the European Union (Centro de Investigación de Galicia accreditation 2019-2022, ref. ED431G2019/01; Consolidation Program of Competitive Reference Groups, ref. ED431C 2021/30).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sang Tae, P.; Jayoung, K. Trends in Next-Generation Sequencing and a new era for Whole Genome Sequencing. *Int. Neurol. J.* **2016**, *20* (Suppl. S2), S76–S83.
2. McCombie, W.R.; McPherson, J.D.; Mardis, E.R. Next-Generation Sequencing technologies. *Cold Spring Harb. Perspect. Med.* **2019**, *9*, a036798. [CrossRef] [PubMed]
3. Chung, W.; Ho, J.; Lin, C.; Lee, D.T. CloudEC: A MapReduce-based algorithm for correcting errors in next-generation sequencing Big Data. In Proceedings of the 2017 IEEE International Conference on Big Data, Boston, MA, USA, 11–14 December 2017; pp. 2836–2842.
4. O'Driscoll, A.; Daugelaite, J.; Sleator, R.D. 'Big data', Hadoop and cloud computing in genomics. *J. Biomed. Inf.* **2013**, *46*, 774–781. [CrossRef] [PubMed]
5. Dean, J.; Ghemawat, S. MapReduce: Simplified data processing on large clusters. *Commun. ACM* **2008**, *51*, 107–113. [CrossRef]
6. Zaharia, M.; Xin, R.S.; Wendell, P.; Das, T.; Armbrust, M.; Dave, A.; Meng, X.; Rosen, J.; Venkataraman, S.; Franklin, M.J.; et al. Apache Spark: A unified engine for Big Data processing. *Commun. ACM* **2016**, *59*, 56–65. [CrossRef]
7. Leinonen, R.; Sugawara, H.; Shumway, M. The Sequence Read Archive. *Nucleic Acids Res.* **2010**, *39* (Suppl. 1), D19–D21. [CrossRef] [PubMed]